Pedro Luis Kantek Garcia Navarro

Computação Evolutiva Aplicada a Problemas Inversos com Preservação da Espacialidade na Representação dos Indivíduos

FLORIANÓPOLIS

2000

UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Computação Evolutiva Aplicada a Problemas Inversos com Preservação da Espacialidade na Representação dos Indivíduos

Tese submetida à Universidade Federal de Santa Catarina
como parte dos requisitos para a obtenção
do grau de Doutor em Engenharia Elétrica

PEDRO LUIS KANTEK GARCIA NAVARRO

Florianópolis, SC, março de 2000

COMPUTAÇÃO EVOLUTIVA APLICADA A PROBLEMAS INVERSOS COM PRESERVAÇÃO DA ESPACIALIDADE NA REPRESENTAÇÃO DOS INDIVÍDUOS

Pedro Luis Kantek Garcia Navarro

Esta Tese foi julgada adequada para obtenção do Título de Doutor em Engenharia Elétrica, Área de Concentração em Sistemas de Informação e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina.

Prof. Guilherme Bittencourt, Dr.
Orientador

Prof. Ildemar Cassana Decker, D.Sc.
Coordenador do Programa de Pós-Graduação em Engenharia Elétrica

Banca Examinadora:

Prof. Guilherme Bittencourt, Dr.
Presidente

Prof. Pedro Paulo Balbi de Oliveira, Dr.
Co-orientador

Prof. Hélio José Correa Barbosa, Dr.

Prof. Christoph Thomalla, Dr.

Prof. Jorge Muniz Barreto, D.Sc.

Prof. Marcelo Stemmer, Dr.

Profª Sandra Aparecida Sandri, Dra.

A Don Yenda e Doña Teté, pela vida
Ao Felipe e à Paula, pela convivência
A Ana Helena, pelo carinho
Ao Pedro Paulo, pelo exemplo

Agradecimentos

A CELEPAR, Companhia de Informática do Paraná, que me manteve ao longo desses anos, sempre apoiando e incentivando a realização do curso. Valem menções especiais ao Paulo Miranda, diretor presidente à época da minha saída para estudar em Florianópolis, e a Maria Alexandra, que era gerente da área de Prospecção Tecnológica, onde então trabalhava. Aos atuais diretores, Francisco Krassuski, Danilo Scalet, Lúcio Hansel e ao gerente de atendimento a clientes, Luiz Ortolani. Referência também ao falecido Rogério Mendes que me acolheu de braços abertos no meu retorno à Gerência de Atendimento a Clientes.

Ao UNICENP, Centro Universitário Positivo, nas pessoas do Prof. Sérgio Lima, ex-diretor e do Prof. Francisco Kantek, coordenador do curso de informática e a PUC-PR, Pontifícia Universidade Católica do Paraná, esta representada pelos professores Dr. Flávio Bortolozzi e Dr. Manuel Camillo, pelo apoio.

Aos meus alunos, companheiros involuntários desta jornada. Boa parte das aulas "truncadas" dos últimos seis anos, tiveram esta causa que ora se encerra.

Ao Governo Brasileiro, através do CNPQ, pela bolsa estudantil que me permitiu viver em Florianópolis durante a fase inicial do curso.

Aos colegas da Celepar, que se desdobraram para me substituir quando minhas atividades de aluno invadiam o espaço reservado ao profissional de sistemas e que também ajudaram nas centenas de horas de máquina vorazmente consumidas pelo modelo aqui estudado. Não posso esquecer da Jane Mary, Elaine Maria, e Jorge Haas, companheiros de todas as horas. A Jane Cherem e Ana Maria, além de tudo substitutas formidáveis.

Ao INPE, Instituto Nacional de Pesquisas Espaciais e UNIVAP, Universidade do Vale da Paraíba, ambos em São José dos Campos, São Paulo, instituições que sempre me receberam com fidalguia, franqueando-me laboratórios, bibliotecas, cantinas e computadores.

Ao pessoal do INPE, em especial aos doutores Fernando e Haroldo que gentilmente compartilharam comigo um pouco de sua experiência e conhecimento. Também referência ao doutor Stephan auxílio indispensável no uso dos computadores do INPE.

As fadas madrinhas de Florianópolis, Salete, Margarida e Isolete, pelo apoio emprestado no início de tudo. Quem tem suporte como este se instala em menos de 24 horas em qualquer canto do mundo.

A Maria José da CELEPAR e Célia do UNICENP e nomeio apenas estas duas para homenagear a todas as bibliotecárias, autênticas bússolas que nos ajudam a sair da escuridão da ignorância sem muitos arranhões.

Aos meus irmãos, Cristina e Carlos pela ajuda inestimável com o inglês e todas as suas complexidades.

A UFSC, Universidade Federal de Santa Catarina, instituição que me acolheu no ano de 1994, bem como ao Curso de Pós Graduação em Engenharia Elétrica - CPGEEL, seus professores e coordenador, e em particular ao antigo LCMI, Laboratório de Controle e Microinformática, atual DAS, Departamento de Automação e Sistemas, também com seus professores e coordenador, doutor Jean Marie Farines. Todas essas pessoas e siglas ficam comigo como exemplos de competência, dedicação e trabalho levado à sério. Vale a citação aos colegas de curso, companheiros de estudos, e por último, mas bem importante, a lembrança ao Wilson e ao Marcos, nossos competentes secretários.

Ao professor Guilherme, que com seu entusiasmo contagiante pelo domínio das "máquinas pensantes" ajudou a estabelecer o contexto da tese orientando nos primeiros e importantes passos. Mais que um orientador, ele foi companheiro todo o tempo.

Finalmente, ao professor Pedro Paulo, a quem devo não apenas a condução geral do trabalho, e por conseqüência o impulso e o estímulo necessários e suficientes à sua conclusão, mas a obtenção de um raro modelo de professor e profissional.

Resumo da Tese apresentada à UFSC como parte dos requisitos necessários para a obtenção do grau de Doutor em Engenharia Elétrica

Computação Evolutiva Aplicada a Problemas Inversos com Preservação da Espacialidade na Representação dos Indivíduos

Pedro Luis Kantek Garcia Navarro

Março de 2000

Orientador: Guilherme Bittencourt

Co-Orientador: Pedro Paulo Balbi de Oliveira

Área de Concentração: Sistemas de Informação

Palavras Chave: computação evolutiva, problemas inversos, inversão magnetotelúrica, operadores evolutivos e *crossover* espacial.

O presente trabalho estuda a utilização de representações espaciais em engenhos de computação evolutiva, como ferramenta para a solução numérica de problemas inversos. Esta classe de problemas apresenta dificuldades para sua solução pela ausência de modelos analíticos, além de eventuais degradações introduzidas por ruído. Uma das formas tradicionais de resolução de problemas inversos usa métodos de otimização. O presente trabalho o faz utilizando um engenho evolutivo no processo de otimização. A tese se inicia descrevendo a computação evolutiva e mostrando algumas técnicas de otimização convencional. Nessa descrição são introduzidos os principais conceitos associados à computação evolutiva em suas várias vertentes: estratégia evolutiva, programação evolutiva e algoritmos genéticos. A seguir são apresentados os problemas inversos utilizados para estudo e para o desenvolvimento da metodologia de seu tratamento: condutividade geomagnética, transmissão de calor e de condutividade térmica. Seguem-se resultados numéricos obtidos para o primeiro deles, na comparação de uma abordagem clássica com regularização versus a otimização evolutiva. Nesta, são também descritos os diversos parâmetros utilizados. Citam-se alguns resultados obtidos para os outros dois problemas. Finalmente, conclui-se afirmando que o método evolutivo quando aplicado aos problemas inversos aqui estudados, nos quais as soluções possuem relações espaciais de vizinhança, é eficaz e robusto quanto aos parâmetros, quanto à inicialização e em relação ao ruído, tendo uma eficiência computacional aceitável.

Abstract of Thesis presented to UFSC as a partial fulfillment of the requirements for the degree of Doctor in Electrical Engineering

Evolutionary Computation Applied to Inverse Problems, with Preservation of Space in the Representation of the Individuals

Pedro Luis Kantek Garcia Navarro

March 2000
Supervisor: Guilherme Bittencourt, PhD
Co-Supervisor: Pedro Paulo Balbi de Oliveira, PhD
Concentration area: Information Systems
Keywords: evolutionary computation, inverse problems, magnetotelluric inversion, genetic operators, spatial crossover.

This thesis addresses the use of spatial representation of individuals in evolutionary computation engines as a tool for a numeric solution for inverse problems. These kinds of problems present difficulties to their solution due to the lack of analytic models and to the degradation entailed by the presence of noise. A traditional approach in inverse problems use optimization mechanisms. Here, an evolutionary engine is used with the same purpose.

The thesis begins with the description of evolutionary computation and showing some standard optimization techniques. In this description, key concepts associated to evolutionary computation and its variants are introduced: evolutionary strategy, evolutionary programming and genetic algorithms. Next, the inverse problems that have been studied are presented, related to magnetotelluric inversion, heat transfer, and thermal condutivity. Subsequently, numerical results are shown, obtained for the first problem, out of the comparison between a classical, regularization-based approach and the evolutionary-based optimizer. In this case, various parameters which were used are also described. Then, the experiments and results obtained for the other two problems are commented upon. Finally, conclusions are drawn that the evolutionary computation method that has been developed, when applied to inverse problems, has acceptable computational efficiency, and is reliable and robust in respect to its parameter setting, the initial condition of the search, and noise presence.

Sumário

Lista de Tabelas

Lista de Figuras

1. Introdução

O estudo de problemas inversos tem grande importância científica e econômica. Diversos fenômenos físicos tem seu tratamento permitido graças a existência de técnicas que habilitam o encontro de soluções para problemas inversos. Estes, tem uma característica que dificulta soluções analíticas exatas, a assim chamada *ill-posedness* que poderia ser traduzida por *problemas mal postos*. Suas soluções deixam de ter pelo menos um dos critérios de existência, unicidade ou continuidade. A característica principal de problemas *ill-posed* é uma instabilidade que surge ao se analizarem dados experimentais. Além do fato de que em dados experimentais sempre está presente algum nível de ruído, o que torna mais instável o problema.

Em geral, não há meio analítico de resolver diretamente problemas inversos. Há que se ter mecanismos de otimização para permitir uma busca iterativa e incremental das soluções para eles. Nesta tese utilizou-se um engenho evolutivo para fazer o papel de otimizador. Houve necessidade de hibridizar este engenho, agregando a ele algum conhecimento dos problemas a resolver. A principal injeção de conhecimento se deu pela utilização de representações que respeitam a vizinhança espacial dos candidatos a solução. Usou-se também um processo de busca local associado aos procedimentos evolutivos.

O mesmo ferramental foi aplicado a 3 problemas inversos distintos e ao final do processo foi possível chegar a algumas conclusões tanto de ordem genérica em relação à computação evolutiva, quanto especificamente em relação aos problemas tratados.

Eis a seguir uma descrição dos capítulos

- Capítulo 2: Apresentação da computação evolutiva (CE). Suas origens, vertentes, motivação inicial e principal (imitar o processo de solução do mais complexo problema já surgido na face da terra: a vida, através da simulação da evolução natural) são apresentadas e discutidas. Segue-se uma inserção da CE na ciência da computação e uma lista incompleta mas representativa dos marcos no seu desenvolvimento. As três principais vertentes da computação evolutiva, suas origens e algoritmos básicos são apresentados. A programação evolutiva, a estratégia evolutiva e os algoritmos genéticos são vistos, iniciando pelo desenvolvimento autônomo de suas idéias e posteriormente, agregando-se como vertentes de uma idéia e um fio condutor mais genérico, mais global e unificado. Vem a seguir uma rápida apresentação das idéias básicas da computação evolutiva, quais sejam: a de população de soluções, com a

conseqüente sugestão forte de processamento paralelo; a de aptidão de cada indivíduo, como atributo individual de cada um; a idéia de codificação de cada indivíduo (em linguajar evolutivo ou genético dir-se-ia genótipo e fenótipo); os operadores de seleção, de recombinação e de mutação; e finalmente, a simulação da passagem das gerações com compressão do tempo, já que o processo é apenas simulado. Uma rápida apresentação de alguns mecanismos mais genéricos de otimização é feita até para permitir comparação com as idéias da CE. São eles: pesquisa randômica, métodos baseados em gradiente, têmpera simulada e métodos compostos. Seguem-se duas formulações genéricas para a computação evolutiva: a algorítmica e a matemática. A primeira descreve como realizar o processamento e a segunda apresenta a idéia, o conceito, por trás dos algoritmos.

- Capítulo 3: Os problemas estudados, iniciando com o conceito de problema inverso e mostrando qual a dificuldade principal na sua solução (*ill-posedness)*. Este fato surge da inexistência de função inversa, obrigando a que se trabalhe com soluções incertas e sempre degradadas. Após, vem a apresentação dos problemas tratados na tese. O primeiro deles, que consumiu cerca de 85% do tempo e dos recursos foi o problema de distribuição de condutividades geoelétricas da terra. O problema direto é a obtenção do campo magnético a partir de um perfil de condutividades conhecido. Este problema (o direto) é solucionado através das equações de eletromagnetismo de Maxwell. Ocorre que a condutividade está no subsolo enquanto que o campo magnético pode ser medido na superfície. Assim, a formulação direta implicaria cavar a terra, medir a condutividade e finalmente calcular o campo magnético que está por toda parte, inclusive na superfície. De maior interesse prático é a formulação inversa: obter a condutividade (no subsolo) a partir da medida de campo magnético (na superfície), sem ser necessário cavar. Este problema já foi estudado e resolvido por Ramos e Campos Velho (1996), utilizando técnicas de regularização e um otimizador da biblioteca *NAG Fortran Library*. Nesta tese, substituiu-se aquele otimizador por em engenho evolutivo. Em ambos os casos, há uma grade de 11 x 8 prismas retangulares que corta a superfície terrestre. Cada prisma tem uma condutividade elétrica que se busca reconstituir. Por outro lado, há um conjunto de 11 pontos na superfície terrestre e em 20 freqüências logaritmicamente espaçadas variando no intervalo entre 0.0001 e 0.01 Hz onde se mediu o campo magnético com as suas componentes real e imaginária. Foi adicionado um ruído gaussiano com taxa de 1%. A partir destas medidas de campo magnético é que se reconstitui a condutividade. Segue-se um problema de transmissão de calor no qual uma placa é construída com

2 materiais de condutividade térmica fornecida (e, no caso, com um material 100 vezes mais condutivo que o outro). O problema direto é obter a condutividade de uma placa formada pelos dois materiais componentes, segundo algum padrão de mistura deles, fornecendo-se a sua constituição. Já o problema inverso é, dada uma condutividade esperada, sugerir qual a constituição da placa. Finalmente, o terceiro e último problema estudado busca reconstituir padrões de defeitos de fabricação em materiais compostos, a partir da observação dos valores de temperaturas observados em uma das faces do material. Neste caso o problema direto é estabelecer os padrões térmicos a partir da disposição de defeitos previamente conhecida. O problema inverso é inferir os padrões de defeitos na composição do material a partir da análise da variação da temperatura em uma das faces do material.

- Capítulo 4: Descrição do processo de estudar e resolver os problemas. Optou-se pelo uso do programa de computação evolutiva GALOPPS (*The Genetic Algorithm Optimized for Portability and Parallelism System*) em função de sua disponibilidade e qualidades que são lá apresentadas. Vem a seguir o processo seguido na solução do problema de reconstrução das distribuições de condutividade geoelétrica, descrevendo-se os problemas encontrados e as soluções adotadas. Mostram-se as diversas funções objetivo utilizadas e a representação cromossômica que foi adotada, bem como justificam-se estas escolhas. Construiu-se uma tabela que apresenta os valores recomendados para os parâmetros (taxa de mutação, taxa de *crossover*, tamanho de população, tipo de *crossover*, tamanho do torneio e número de gerações) e os valores que foram usados. Uma ligeira apreciação sobre o uso de multipopulações, funções de convergência forçada, existência de simetria e possibilidade de especialização crescente foram incluídos. Concluiu-se esta parte respondendo à questão "porque é necessário hibridizar".
- Capítulo 5: Mostram-se os resultados alcançados, para o problema geofísico (ou geomagnético). Após ligeira introdução, apresenta-se o arcabouço usado na solução. Seguem-se as 5 configurações de condutividade que foram usadas para testar e validar o modelo e uma lista dos valores empregados para os parâmetros em cada execução. Por causa da necessidade de hibridização, criaram-se operadores específicos ao problema. O primeiro, denominado SPAUC (*spatial uniform crossover*), é uma generalização da recombinação uniforme, na qual é introduzida a noção de vizinhança espacial. O próximo operador *ad-hoc* é uma busca local que utiliza uma subida da encosta (*hill climbing*) bastante simples. Na continuidade, o trabalho descreve o operador de homogeneização. Ele está baseado (e justificado) na alta pro-

babilidade de que regiões vizinhas tenham condutividades próximas ou iguais. O operador é descrito em termos de seus processos, componentes e parâmetros. A próxima etapa compara o método evolutivo versus uma abordagem que utiliza um otimizador convencional associado a uma técnica nova de regularização. São feitas diversas execuções variando-se os parâmetros da regularização de primeira e de segunda ordem, respectivamente γ_0 e γ_1 para o engenho clássico e outras tantas para o engenho evolutivo variando-se a semente aleatória, já que este segundo processo é estocástico. O teste seguinte visa determinar a degradação introduzida no resultado quando o nível de ruído cresce. São feitos testes com 2%, 3%, 4% e 8% de ruído gaussiano, tanto no engenho clássico quanto no evolutivo. Depois, vem a comparação de uma configuração mais irregular que as outras.

- Capítulo 6: Descrição dos resultados obtidos no estudo de outros dois problemas inversos, ambos relacionados a fenômenos de transmissão de calor: o primeiro estudou a construção de placas de 2 materiais distintos, sujeito a restrições de ordem tecnológica e denominado "problema de transmissão de calor" e o segundo um problema de localização de impurezas, também em materiais compostos, a partir do comportamento térmico deste material. Este problema foi denominado "difusão de calor em materiais compostos". Usou-se para ambos o mesmo ferramental – devidamente adaptado – do primeiro problema com resultados satisfatórios.
- O capítulo 7, intitulado "Idéias para uma metodologia" descreve uma proposta de alguns passos metodológicos necessários para atacar e resolver um problema inverso usando a computação evolutiva.
- Finalmente, o capítulo 8, "Conclusões" sumariza o trabalho executado e estabelece as bases para a continuação do trabalho. Concluiu-se pela necessidade do uso de operadores híbridos e pela eficácia da abordagem integrando um engenho evolutivo associado a processos de busca local. Mostrou-se que esta técnica é robusta em relação ao estabelecimento de parâmetros, à presença de ruído e à inicialização irregular. Finalmente, em termos de eficiência computacional quando comparada com outras técnicas de solução de problemas inversos, a aqui apresentada mostrou um consumo maior, mas ainda assim aceitável, de recursos computacionais. Na continuação deste estudo, sugere-se a busca de melhores critérios de parada (até para melhorar a eficiência computacional do processo), o incremento da técnica para tratamento de problemas de alta dimensionalidade, e o aprofundamento do estudos dos operadores evolutivos aqui apresentados.

A principal contribuição da tese é a de propor a solução de problemas inversos nos quais as soluções possuem relações espaciais de vizinhança usando computação evolutiva. A apresentação, lado a lado de resultados clássicos e evolutivos para o mesmo problema também merece menção. Finalmente, o uso de operadores evolutivos especialmente desenvolvidos bem como a apresentação de seus desempenhos é o ponto chave da tese.

2. Computação Evolutiva

2.1 Vocação

Começa-se citando Goldberg (1989) que escreveu "*algoritmos evolutivos podem ser aplicados a virtualmente qualquer problema*". Entretanto, olhando sob um prisma um pouco mais restrito, pode-se afirmar que a CE tem a habilidade de manusear problemas complexos. Mesmo sem oferecer respostas absolutamente corretas (até pela complexidade dos problemas), ela sempre tem algo a oferecer na busca de resultados para este tipo de questão. Mais especificamente ainda falando, como saber se para um dado problema a computação evolutiva tem algo a oferecer ?

Não há resposta definitiva, mas parece haver consenso na literatura sobre o tema de que problemas com algumas das seguintes características:

- espaço de busca muito grande (o que inviabiliza a busca exaustiva)
- espaço de busca não completamente conhecido
- função de avaliação sujeita a ruídos ambientais, ou não completamente conhecida
- não necessidade da obtenção de um máximo global, sendo satisfatório um máximo local ainda que sujeito a algum critério de aceitação
- espaço de busca multimodal

poderiam ser satisfatoriamente abordados usando ferramentas da CE. Note-se que este enfoque pressupõe CE "pura", sem hibridização. A partir do instante que se agrega a esta abordagem um conhecimento advindo do problema, pode-se generalizar a vocação da CE, e fazer coro a Goldberg (1989) quando este diz que qualquer problema pode ser tratado usando CE. Restaria apenas a questão de "quanto" do conhecimento do problema deve ser agregado.

2.2 Conceituação

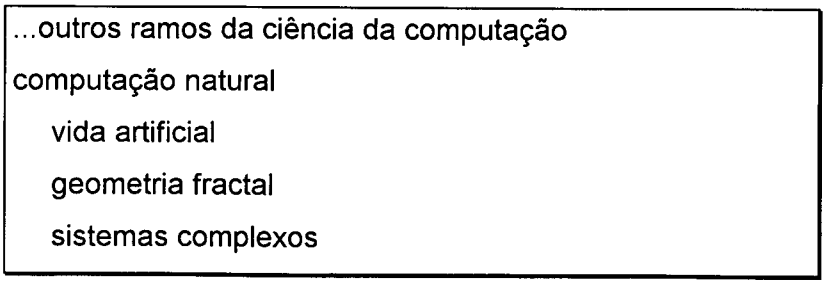
A computação evolutiva é um ramo da ciência da computação que propõe um paradigma alternativo ao processamento de dados convencional. Enquanto neste, antes de se lançar mão de um computador para resolver ou auxiliar na solução de uma deter-

minada questão, há que se conhecer previamente a maneira de encontrar a solução, na computação evolutiva este conhecimento pode ser prescindido.

Na CE (computação evolutiva), tal conhecimento é substituído por uma simulação do mecanismo da evolução natural. Como já descreveu Darwin, a vida em nosso planeta é o resultado de um processo aparentemente aleatório, no qual o meio ambiente exerceu e exerce um papel de modelador e de árbitro do resultado. A constatação da diversidade da vida, associada ao fato de que todos os seres vivos compartilham uma bagagem genética comum, pelo menos em termos de seus componentes básicos, fornece bem uma idéia de quão rica e multifacetada tem sido a atuação do meio ambiente na condução do processo de manutenção e melhoria da vida. A CE está baseada em quatro processos fundamentais: reprodução, variação randômica, competição e seleção de indivíduos dentro da população. Segundo Atmar, citado em Fogel (1997a), *"eles formam a essência da evolução, e sempre que estes quatro processos surgem, seja dentro de um computador, seja na natureza, a evolução inevitavelmente aparece".*

A localização da CE no espectro da ciência da computação ainda é controvertida, provavelmente pelo pouco tempo decorrido desde a publicação de suas idéias fundamentais. Mais ainda, tal ciência não teve origem única, surgindo variantes ao longo dos anos 60-80, em situações diferentes, em países diferentes e para tentar resolver problemas diferentes. Só agora, no final dos anos 90, é que se assiste uma tentativa de unificação de tais variantes, buscando-se uma dogmatização que consolide e selecione as principais idéias do campo.

Possivelmente, a melhor inserção da CE na ciência da computação seja aquela descrita em (Heitkoetter e Beasley, 1999), a qual deve ser vista, não como posição definitiva (que de resto não existe ainda), mas apenas como uma proposta de classificação. Segundo Heitkoetter, o maior ramo de classificação que abrange a CE é o que foi chamado de "Computação Natural". Ele inclui os tópicos de vida artificial, geometria fractal, sistemas complexos e um ramo que foi identificado como inteligência computacional. Este último inclui redes neurais artificiais, sistemas nebulosos e a computação evolutiva. Na figura 1, tem-se o aspecto visual desta proposta de classificação



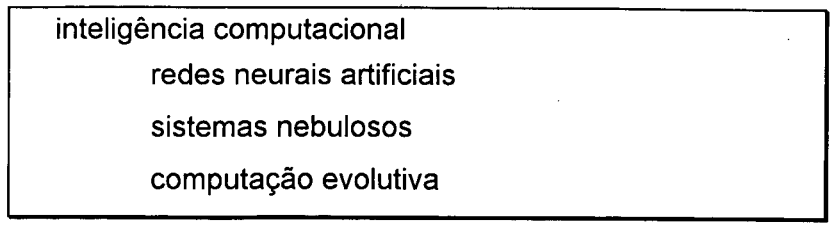


Fig. 1: Localização da CE dentro da ciência da computação, segundo Heitkoetter (1999).

O próprio termo "computação evolutiva" é novo. Segundo Fogel, (1997a), ele começa a ser usado apenas a partir de 1991. A CE está baseada em algumas idéias básicas, que quando implementadas, permitem simular o processo de passagem de gerações da evolução natural dentro de um computador. Pode-se discutir se a seleção natural tem ou não um objetivo, se o universo e os seres vivos fazem parte de um plano ou são mero acaso, discussão que não cabe aqui, mas se a seleção natural tivesse um "objetivo", este poderia ser resumido como a expansão e melhoria da vida. Na CE tal objetivo deve ser substituído por aquele que for mais adequado ao problema que se está a estudar. Feita essa mudança de enfoque, as idéias que permitem a simulação de que se fala acima são:

- A criação de uma população de soluções, possivelmente obtida na sua primeira geração de modo aleatório, e na qual os indivíduos tenham registrado de modo intrínseco os parâmetros para obter uma solução ao problema posto.

- A criação de uma entidade que possa fazer julgamentos quanto à aptidão de cada um dos indivíduos. Essa entidade pode assumir as mais diversas formas (um conjunto de funções matemáticas, um programa de computador, um avaliador de desempenho, entre outros), e ela não precisa deter conhecimento de como encontrar uma solução para o problema, mas apenas de atribuir uma "nota" ao desempenho de cada um dos indivíduos da população.

- E finalmente, mas não menos importante, a criação de uma série de operadores que serão aplicados à população de uma dada geração para obter os indivíduos da próxima geração. Tal série de operadores, é copiada (pelo menos em sua concepção filosófica) da natureza. Os principais operadores citados na literatura são:

- seleção: permite escolher um indivíduo ou um par deles para gerar descendência. Note-se que este operador simula a reprodução assexuada (no primeiro caso) e a sexuada (no segundo) que ocorrem na natureza. Obviamente, a prioridade da escolha recai sobre indivíduos mais bem avaliados pela entidade de avaliação. Note-se que a seleção não cria novas soluções, apenas indica, das existentes, qual (is) a (s) melhor (es).

- recombinação: operador que simula a troca de material genético entre os ancestrais para a geração da prole. Basicamente o conceito é fácil de entender: se há dois indivíduos que são bem avaliados, embora por duas razões distintas, a recombinação poderia, em tese, criar um descendente que juntasse essas duas razões em um único indivíduo que seria melhor que seus ancestrais.

- mutação: operador que introduz mudanças aleatórias sobre o material genético.

A CE se divide em seus diversos (sub)campos de acordo com a importância e a operacionalização dos conceitos vistos acima, quando de sua implementação em modelos computacionais. A figura 2, mostra uma possível distribuição de tais subcampos.

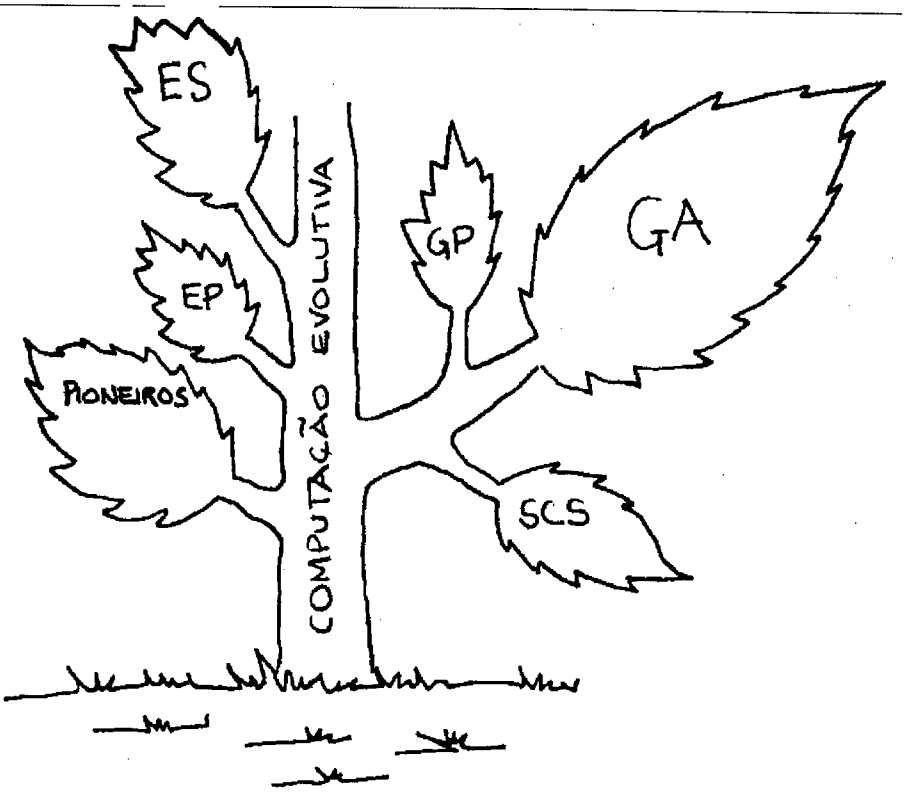


Fig. 2: A Computação Evolutiva e seus campos

Cada uma das folhas desta árvore representa uma tendência ou uma concepção prévia a respeito de algumas características que a simulação da evolução natural deve ter. Historicamente, as primeiras iniciativas na área (representadas pela folha chamada de "pioneiros"), são de biólogos e geneticistas interessados em simular os processos vitais em computador, o que recebeu na época o nome de "processos genéticos". Alguns desses cientistas citados em Goldberg (1989) são: Barricelli, 1957, 1962; Fraser, 1960, 1962; Martin e Cockerham, 1960. Um biólogo, Rosenberg, em 1967 ao escrever sua tese de doutorado, simulou uma população de seres unicelulares, estrutura genética clássica (um gene, uma enzima), com estrutura diplóide, com cromossomos de 20 genes e 16 alelos permitidos em cada um (Goldberg, 1989).

Já na década de 60, Holland e outros começaram a estudar os chamados sistemas adaptativos, que foram modelados como sistemas de aprendizagem de máquina. Tais modelos conhecidos como algoritmos genéticos (GA na figura 2) implementavam populações de indivíduos contendo um genótipo, formado por cromossomos (que neste modelo eram representados por seqüências de bits), e aos quais se aplicavam os operadores acima citados: seleção, recombinação e mutação. Ainda que Holland tenha pro-

posto um quarto operador (a inversão), este não chegou a ser largamente usado (Davis, 1991).

Uma das primeiras aplicações propostas para os algoritmos genéticos (cuja primazia no uso do termo cabe a Bagley na sua dissertação em 1967), seguindo o caminho pesquisado por Holland, foram os sistemas classificadores (identificados na figura 2 pela folha contendo SCS). Tais aparatos são sistemas de produção, e na verdade usam os algoritmos genéticos em uma parte do algoritmo global. A despeito do interesse que levantaram na época, os sistemas classificadores permanecem como um campo de estudo ainda inexplorado. Nas palavras de Heitkoetter citando Goldberg, "sistemas classificadores são um pântano. Um maravilhoso e inventivo pântano, mas ainda assim, um pântano (Heitkoetter, 1999).

Outro ramo descendente dos algoritmos genéticos é o da programação genética. Aqui, os indivíduos da população não são seqüências de bits, mas sim programas de computador, armazenados na forma de árvores sintáticas. Tais programas é que são os candidatos à solução do problema proposto. Programação genética não usa o operador mutação e a recombinação se dá pela troca de sub-árvores entre dois indivíduos candidatos à solução.

A seguir tem-se a programação evolutiva, na qual se busca predizer o comportamento de máquinas de estado finitas. Apenas dois operadores foram originalmente usados: a seleção e a mutação. As idéias datam de 1966, e não foram muito consideradas na comunidade na computação evolutiva por rejeitar o papel fundamental da recombinação, embora esta abordagem esteja sendo reavaliada.

Finalmente, completa a árvore a estratégia evolutiva, proposta em meados dos anos 60, na Alemanha. A ênfase aqui é na auto-adaptação. O papel da recombinação é aceito, mas como operador secundário. Segundo Goldberg, "a estratégia evolutiva teve devotos em certos círculos científicos e de engenheiros, particularmente na Alemanha (Goldberg, 1989).

Em 1990, a comunidade da computação evolutiva reuniu esforços para a realização do primeiro evento envolvendo todas as iniciativas. Tratou-se do *Workshop on Parallel Problem Solving from Nature*, que ocorreu em Dortmund. Este congresso foi sucessivamente realizado, e diversos outros surgiram. Uma medida do interesse crescente deste ramo de estudo e do seu rápido desenvolvimento pode ser observado na *newsletter* eletrônica *Genetic Algorithms Digest,* de 17 de fevereiro de 99, (Volume 13 : *Issue* 4), apresenta nada menos que 32 chamadas de congressos ou eventos científicos diretamente relacionados ao tema.

Note-se que a despeito de suas origens bastante diversas, é impossível deixar de mencionar quanto todas estas abordagens têm em comum. Desde o modelo filosófico conceitual inicial (a evolução natural), passando pelo conjunto de operadores e principalmente tendo em vista o objetivo final de tais sistemas, que é o da solução de problemas complexos, todos esses ramos têm muito mais semelhanças do que diferenças. Graças a isso é razoável prever que no futuro, haverá uma unificação da teoria, possivelmente mantendo-se cada uma das alternativas exatamente no papel de alternativas de uma abordagem global.

De um modo geral, pode-se dizer que os algoritmos evolutivos representam uma abordagem distinta dos métodos tradicionais, no mínimo pelas seguintes razões (Golberg, 1989, modificado)

- trabalham com parâmetros codificados e não diretamente sobre os parâmetros;
- trabalham com uma coleção de pontos candidatos a solução e não apenas com um ponto;
- usam uma função objetivo para pontuar os candidatos, prescindindo desta forma, qualquer conhecimento prévio do problema;
- usam regras de transição probabilísticas.

É uma hipótese a de que a robustez dos métodos evolutivos se deve a estas características, embora isto tenha sido explicitamente afirmado por Goldberg (1989). Schwefel (1997), diz: "*É muitas vezes admirável o fato de que mesmo estabelecendo parâmetros obviamente ruins, não se consegue evitar bons resultados*". Isto certamente pode ser descrito como robustez. Citando novamente o mesmo autor, pode-se concordar com "*a melhor coisa que se pode dizer sobre os algoritmos evolutivos é que eles apresentam um arcabouço metodológico que é fácil de entender e usar, que pode ser usado na modalidade de caixa preta, ou ser aberto para incorporar receitas novas ou antigas, que incrementem a sofisticação, especialização ou a hibridização. Pode ser usado em situações dinâmicas, onde o objetivo ou as restrições mudam ao longo do tempo, onde o ajuste de parâmetros ou medidas de desempenho são distorcidas por ruído, ou onde o espaço de soluções é irregular, descontínuo, multimodal, fractal ou de alguma maneira não pode ser atacado pelas ferramentas tradicionais, especialmente quando elas necessitam de predições globais a partir de análises da superfície local*".

2.3 Implementação

> *"Indivíduos e espécies podem ser vistos como uma dualidade entre seus programas genéticos – chamado genótipo, e sua expressão em forma de traços de comportamento, o fenótipo."* (Fogel, 1997b).

O conceito chave na computação evolutiva, como definido por Holland é o da "*adaptação (ad aptare, ou tornar-se apto a) o que permite usar a computação evolutiva em grande classe de problemas distintos, e unificar a abordagem de sua solução*" (Holland, 1992b, modificado).

Para essa classe de métodos de solução, uma população de soluções é usualmente gerada de maneira randômica e evolui ao longo das gerações que são simuladas no processo, em direção às soluções de maior valor da função objetivo, por meio de processos de seleção, mutação e recombinação.

Denomina-se, na computação evolutiva, a função objetivo como "aptidão" (em inglês "fitness"). A função objetiva pode ser tão sofisticada quanto se queira (ou possa), mas no caso mais simples é uma simples decisão binária (a solução *a* é melhor ou pior do que a solução *b* ?). O processo de seleção busca indivíduos com alta aptidão para formar descendência, à qual são aplicados os outros dois processos. O primeiro, indica uma possível modificação aleatória na solução, enquanto o segundo significa a troca de partes da solução entre dois ancestrais gerando dois descendentes.

O conjunto inicial de soluções pode ser aleatório ou pode ser obtido a partir das técnicas convencionais que já são dominadas para resolver instâncias mais simples do problema que está sendo tratado. De um lado, usando-se soluções inicialmente aleatórias, pode-se usar sempre o mesmo algoritmo, os mesmos operadores, e socorrer-se da teoria matemática preexistente, como por exemplo, o teorema dos esquemas (Goldberg, 1989). Por outro lado, adaptando o conceito de computação evolutiva a um problema específico e empregando soluções iniciais obteníveis por métodos convencionais, há um trabalho mais pesado ao ser necessário adaptar os operadores usuais da computação evolutiva para o problema específico, mas em compensação, na pior das hipóteses, a solução encontrada é igual à melhor solução obtida anteriormente pelas técnicas convencionais. Qualquer melhoria que venha a ser obtida pelo uso da computação evolutiva representará benefício.

Além da função de avaliação, é necessário encontrar uma maneira de codificar as soluções para o problema que se quer resolver. O resultado dessa codificação fará o papel dos cromossomos na evolução natural. A partir desses cromossomos é que a função objetivo deve ser capaz de se manifestar e registrar quão boa é aquela solução. Denomina-se genótipo a este conjunto de cromossomos.

Para gerar novas soluções, usam-se os operadores evolutivos, que se aplicam a um indivíduo (solução) ou a pares delas. Tudo se dá como houvesse uma descendência monoparental, chamada assexuada, na natureza, ou cruzamento entre duas soluções, gerando descendência, denominada sexuada, na natureza. Tal processo repetido inúmeras vezes simula a passagem de gerações e gerações de seres vivos no mundo biológico. Como o processo está apenas sendo simulado em um computador digital, o fator tempo pode ser comprimido sem perda de qualidade.

Estabelecido um conjunto de soluções -- que passarão a funcionar como ascendentes as novas soluções ou os descendentes, obtidas a partir daquelas, sofrem a ação dos chamados operadores evolutivos. Mediante estes operadores, os descendentes passarão a ser diferentes dos ascendentes. Se eles forem melhores, assim julgados pela função de avaliação, terão uma descendência maior, do que se eles forem julgados pouco aptos. A troca de material genético, chamada de recombinação, implica que um par de ascendentes dará origem a um par de descendentes, e cada descendente herdará partes aleatoriamente escolhidas de cada ascendente. A mutação significa a mudança também aleatória de uma parte da solução. No caso mais simples de cromossomos codificados em binário, a mutação seria a simples inversão de um bit. Tanto a recombinação quanto a mutação tendem a ocorrer segundo probabilidades dadas que são parâmetros da técnica.

2.4 Técnicas de Otimização

A solução de um problema, como posto nesta tese deverá ser entendida como a busca da otimização do sistema sob análise. Ou seja, a procura de uma solução, preferencialmente uma boa solução e idealmente a melhor solução. Os problemas estudados são os mais diversos e variados, compartilhando uma única característica: a de não serem triviais ou terem fácil solução.

Como diz Schwefel (1995), *a não existência de método universal de otimização fez com que surgissem inúmeros procedimentos, cada um tendo aplicação limitada apenas a casos especiais.* Boa parte das técnicas de otimização, possuem um escopo reduzido a uma classe de problemas, e em geral, quanto mais complexa é a solução, mais

especializada e conseqüentemente menos geral é o método de solução, quando este existe.

A seguir, uma rápida visão dos principais métodos existentes:

2.4.1 Baseados em cálculo

A solução é encontrada pela resolução do sistema de equações que descrevem o fenômeno em estudo. Quando tal sistema existe e é solúvel, este é o método ideal por excelência, já que é rápido, simples e exato.

2.4.2 Pesquisa randômica

Pontos no espaço de pesquisa são aleatoriamente gerados, e para cada um a função que está sendo otimizada é calculada. Ao final da execução, o ponto que tiver obtido melhor desempenho é selecionado como solução.

Para problemas de menor porte, este processo poderia ser aplicado a todos os pontos do espaço. Nesse caso, o método se denomina pesquisa enumerada, e também nesse caso a solução encontrada é a melhor existente, a despeito do considerável esforço computacional empregado. Este método também é conhecido como *"força bruta"*, e raramente empregado (Beasley, Bull e Martin,1993a)

2.4.3 Métodos do gradiente

Uma classe de métodos de solução foi desenvolvida com base no uso do gradiente da função objetivo em um ponto considerado para guiar a direção de busca. Em linhas gerais, um ou vários pontos são randomicamente gerados, e iniciam uma iteração na qual o próximo ponto é localizado obedecendo a direção dada pelo gradiente da função no ponto anterior.

Este método apresenta duas deficiências importantes. Funções que apresentem descontinuidades não podem ser usadas (pela impossibilidade do cálculo das derivadas necessárias para obter o gradiente) e funções multimodais, pela possibilidade do método se fixar em um máximo local. Estes métodos são conhecidos genericamente pelo nome de *"hill climbing"*, ou em português, "subida da encosta".

2.4.4 Pesquisa iterativa

É uma combinação dos dois métodos acima. O método do gradiente é usado para encontrar um máximo. Em vez de encerrar a pesquisa, um novo ponto é gerado (de maneira randômica) e uma nova pesquisa de gradiente é iniciada. Ao final, o ponto com maior valor da função objetivo é a solução.

Esse método pode funcionar bem para funções que tenham pontos de máximo da função objetivo rodeados por pontos onde tal função tem valor mínimo. Segundo Beasley, Bull e Martin, "este tipo de função é difícil de otimizar por qualquer método e aqui a simplicidade da busca interativa geralmente *wins the day* (no original)" (1993a).

2.4.5 Têmpera simulada

Também conhecida como *simulated annealing.* Variação do método da subida da encosta, na qual no início do processo são feitos movimentos que podem piorar o resultado numérico da função objetivo. A idéia é explorar todo o espaço do problema logo no início a fim de ficar independente do estado inicial.

Até pelo nome, este método baseia-se no processo de resfriamento lento de materiais fundidos. Estes são aquecidos a altos níveis de temperatura, e depois gradualmente resfriados até o estado sólido. O objetivo é produzir um estado com mínimo de energia. Em geral os materiais diminuem sua temperatura, mas existe certa probabilidade de ocorrer uma transição a um estado de maior energia. Esta probabilidade é fornecida por $p = e^{-(\Delta E/kT)}$ onde ΔE é a mudança positiva no nível de energia, T é a temperatura e k é a constante de Boltzmann. Assim, no resfriamento, a probabilidade de um grande movimento ascendente é menor do que a probabilidade de um movimento pequeno. A probabilidade do movimento ascendente diminui a medida em que diminui a temperatura.

O processo todo é dependente de como T diminui. Se T diminuir rapidamente formar-se-ão locais estáveis de alta energia (o que pode ser considerado como um mínimo local na linguagem de otimização de funções). Se o tempo em que T varia for maior, provavelmente será encontrado um mínimo global.

Transportando a analogia para a computação, ΔE não representa mais a mudança de energia mas a mudança no valor da função objetivo, seja ela qual for. No mundo computacional k não tem sentido. Então escolhe-se E e T para que trabalhem bem na forma algorítmica. A formula revisada da probabilidade passa a ser

$p' = e^{-(\Delta E/T)}$.

O algoritmo de têmpera simulada é um pouco diferente da subida da montanha simples. A principal diferença é que movimentos que pioram a função objetivo podem ser aceitos. Segundo Fogel, *"técnicas clássicas, como gradiente, subida da montanha determinística e pesquisa randômica pura, têm se mostrado em geral insatisfatórias especialmente quando aplicadas a problemas de otimização não lineares, especialmente aqueles com componentes estocásticos, temporais ou caóticos"* (Fogel, 1997a)

2.5 Formalização algorítmica

A seguir uma boa formalização algorítmica para a computação evolutiva, extraída de (Bäck, 93). Seja $f:\Re^n \rightarrow \Re$ a função objetivo a ser otimizada e $\Phi : I \rightarrow \Re$ a função de aptidão, onde I é o espaço de indivíduos. $\vec{a} \in I$ denota um indivíduo enquanto $\vec{x} \in \Re^n$ indica as coordenadas de um ponto no espaço de soluções. $\mu \geq 1$ denota o tamanho da população de ancestrais, enquanto $\lambda \geq 1$ é o tamanho da população de descendentes, ou seja o número de indivíduos criados através de recombinação e mutação a cada geração. Se o tempo de vida dos indivíduos está limitado a uma geração, pode-se assumir $\lambda > \mu$. A população na geração t, $P(t) = \{\vec{a}_1(t),\ldots,\vec{a}_\mu(t)\}$ consiste de indivíduos $\vec{a}_1(t) \in I$. $r_{\Theta_r} : I^\mu \rightarrow I^\lambda$ denota o operador de recombinação, o qual pode ser controlado pelo conjunto de parâmetros sumariados pelo conjunto Θ_r. Similarmente, o operador mutação $m_{\Theta_m} : I^\lambda \rightarrow I^\lambda$ modifica a descendência controlado pelos parâmetros Θ_m. A seleção $s_{\Theta_s} : (I^\lambda \cup I^{\mu+\lambda}) \rightarrow I^\mu$ é aplicada na escolha da população de ascendentes para a próxima geração. Durante a etapa de avaliação, a função de aptidão $\Phi : I \rightarrow \Re$ é calculada para todos os elementos da população e $\iota : I^\mu \rightarrow \{verdadeiro, falso\}$ é o critério de fim. Finalmente, $Q \in \{\phi, P(t)\}$ é o conjunto de indivíduos que são tomados adicionalmente durante a etapa de seleção. Se Q é vazio, os indivíduos de uma geração não são usados como candidatos a ascendentes na próxima geração. Se Q for igual a P(t), o contrário ocorre.

Com essa notação, pode-se descrever o algoritmo básico da computação evolutiva como

$t := 0;$

$inicializacao\ P(0) := \{\vec{a}_1(0),\ldots,\vec{a}_\mu(0)\} \in I^\mu;$

$avaliacao\ P(0) :\{\Phi(\vec{a}_1(0)),\ldots,\Phi(\vec{a}_\mu(0))\};$

$enquanto\ (\iota(P(t)) \neq verdadeiro)\ faca$

$recombinacao\ P'(t) := r_{\Theta_r}(P(t));$

$mutacao\ P''(t) := m_{\Theta_m}(P'(t));$

$avaliacao\ P''(t) :\{\Phi(\vec{a}_1''(t)),\ldots,\Phi(\vec{a}_\lambda''(t))\};$

$selecao\ P(t+1) := s_{\Theta_s}(P''(t) \cup Q);$

$t := t+1;$

$fim - faca$

2.6 Formalização matemática

Segue-se uma definição genérica de um algoritmo evolutivo (Bäck, 1996). Um algoritmo evolutivo genérico é definido como uma 8-upla:

$$AE = (I, \Phi, \Omega, \Psi, s, \iota, \mu, \lambda)$$

Onde $I = A_x \times A_s$ é o espaço de indivíduos e A_x, A_s denotam conjuntos arbitrários. $\Phi: I \to \Re$ denota a função de aptidão (fitness) que assinala valores reais aos indivíduos. $\Omega = \{\omega_{\Theta 1},\ldots,\omega_{\Theta z} | \omega_{\Theta i}: I^\lambda \to I^\lambda\} \bigcup \{\omega_{\Theta 0}: I^\mu \to I^\lambda\}$ é um conjunto de operadores genéticos probabilísticos $\omega_{\Theta i}$ cada um dos quais é controlado por parâmetros específicos sumarizados pelos conjuntos $\Theta i \subset \Re$.

$s_{\Theta s}: (I^\lambda \cup I^{\mu+\lambda}) \to I^\mu$ denota o operador seleção, o qual pode modificar o número de indivíduos de λ ou $\lambda+\mu$ até μ, onde μ e $\lambda \in N$ e $\mu = \lambda$ é permitido. Um conjunto adicional Θs de parâmetros pode ser usado pelo operador de seleção. μ é o número de indivíduos ancestrais e λ é o número de indivíduos descendentes.

$\iota: I^\mu \to \{verdadeiro, falso\}$ é um critério para o fim do algoritmo e a função de transição de gerações $\Psi: I^\mu \to I^\mu$ descreve o processo completo de transformação de uma população P em sua geração subseqüente pela aplicação dos operadores genéticos e de seleção

$$\Psi = s \circ \omega_{\Theta i1} \circ \ldots \circ \omega_{\Theta ij} \circ \omega_{\Theta 0}$$
$$\Psi(P) = s_{\Theta s}(Q \cup \omega_{\Theta i1}(\ldots(\omega_{\Theta ij}(\omega_{\Theta 0}(P)))\ldots))$$
$$\{i_1, \ldots, i_j\} \subseteq \{1, \ldots, z\}$$
$$Q \in \{0, P\}$$

2.7 Principais vertentes

Dentro da computação evolutiva existem diversas correntes ou abordagens. Todas compartilham a simulação do processo de seleção natural, mas se diferenciam pela origem e por suas estratégias de solução. Os algoritmos genéticos enfatizam o operador de *crossover* como o mais importante e aplicam a mutação como operador secundário. Advogam um operador de seleção probabilístico (usualmente a roda da roleta) e sugerem representação binária para os indivíduos. Algoritmos de estratégia evolutiva usam mutação com distribuição gaussiana (normal) para modificar vetores de valores reais. Atribuem idêntica importância à mutação e à recombinação. O operador de seleção é determinístico e as populações de ascendentes e descendentes usualmente diferem em tamanho. A programação evolutiva, enfatiza a mutação e não prevê nenhum papel para a recombinação. Ela estende o processo evolutivo aos parâmetros da técnica.

2.7.1 Estratégia evolutiva (ES)

Surgiu a partir de 1964 na Universidade Técnica de Berlim, Alemanha. O problema original em estudo era o de encontrar formas otimizadas para objetos inseridos em fluxos de vento. Estratégias usando o método do gradiente não foram bem sucedidas. Dois estudantes, Ingo Rechenberg e Hans-Paul Schwefel tiveram a idéia de efetuar alterações randômicas nos parâmetros que definiam a forma do objeto, baseado na idéia da seleção natural. *"Rechenberg desenvolveu a teoria da velocidade de convergência para o mecanismo (1+1)-ES, um esquema simples de seleção-mutação trabalhando em um único indivíduo que gera um único descendente por geração através da mutação Gaussiana, e propôs uma regra teórica para controlar o desvio padrão da mutação, conhecida como regra de sucesso 1/5"* (Bäck, 1993). Mais tarde, esta teoria evoluiu para o chamado mecanismo (μ+1)-ES, no qual uma população μ de indivíduos se recombina de maneira randômica para formar um descendente, o qual após sofrer mutação substitui (se for o caso) o pior elemento da população. Ainda que este mecanismo nunca tenha sido largamente usado ele permitiu a transição para os chamados (μ+λ)-ES e (μ,λ)-ES já no

final dos anos 70. Na primeira, os μ ancestrais e os λ descendentes convivem enquanto na segunda, os μ ancestrais "morrem", deixando apenas os λ descendentes "vivos".

A abordagem ES é adequada a uma vasta gama de problemas de otimização, porque ela não necessita de muitas informações sobre o problema (particularmente derivadas da função objetivo). Ela é capaz de resolver problemas multidimensionais, multimodais e não lineares sujeitos a restrições lineares ou não lineares (Heitkoetter, 1999).

2.7.1.1 Algoritmo básico

O algoritmo básico conforme Schwefel (1995), é:

> Etapa 0: Uma dada população consiste de μ indivíduos. Cada um é caracterizado pelo seu genótipo consistindo de n genes o qual determina de modo não ambíguo a aptidão para a sobrevivência.
>
> Etapa 1: Cada indivíduo da população produz λ / μ descendentes, na média, de modo que um total de λ indivíduos novos são gerados. O genótipo dos descendentes difere ligeiramente do de seus ancestrais.
>
> Etapa 2: Apenas os μ melhores indivíduos dos λ gerados permanece e eles tornam-se os ancestrais na próxima geração.

2.7.1.2 Estratégia 1+1

Na estratégia (1+1), uma solução gera outra a cada geração. Nessa operação é aplicada uma mutação normal (ou seja, pequenas alterações têm maior probabilidade de ocorrer do que grandes alterações, seguindo a distribuição normal), até que o descendente tenha um desempenho melhor que seu ascendente, quando então ele lhe toma o lugar.

Na estratégia evolutiva, um indivíduo corresponde a um ponto no espaço de soluções, e possui um genótipo formado por variáveis objetivo e variáveis estratégicas. As variáveis objetivo são aquelas que, sofrendo recombinação e mutação permitem incrementar a aptidão dos indivíduos em direção ao máximo global de otimização. As variáveis estratégicas representam variâncias e co-variâncias, que devem ser operadas junto às variáveis de controle para produzir mutações. Bäck e Schwefel dizem que "freqüentemente, entretanto, apenas as variâncias são tomadas em conta, e em algumas vezes,

é adequado trabalhar com apenas uma variância, válida para todas as variáveis objeto (Bäck, 1993).

Devido à relativa simplicidade deste esquema, é que a regra do 1/5 pode ser aplicada (Heitkoetter, 1999). Esta regra diz que quando a taxa de sucesso (isto é, o descendente tem aptidão maior do que o ascendente), após mutação é maior do que 0,2 a variância da distribuição normal deve ser aumentada, enquanto se ela for menor do que 0,2 a variância deve ser diminuída (Schwefel, 1995).[1] Registre-se que cada indivíduo detém a sua própria variância e eventualmente co-variância (que são os parâmetros básicos da mutação), o que caracteriza o conceito de auto-adaptação.

2.7.1.3 Estratégias soma e vírgula

As estratégias $(\mu+\lambda)$ e (μ,λ), são chamadas estratégia soma e estratégia vírgula respectivamente. Na estratégia soma, os ascendentes são levados em conta durante a etapa de seleção, enquanto na estratégia vírgula, apenas os descendentes de uma dada geração são candidatos a serem selecionados para gerar a próxima. Esta característica dos ancestrais competirem junto com os descendentes frente ao operador seleção é chamada de *elitismo*.

A escolha adequada de μ / λ determina a velocidade de convergência da estratégia evolutiva. Heitkoetter, diz "se o que se busca é uma rápida, ainda que local, convergência, pode-se escolher uma razão de seleção pequena, por exemplo, (5,100). Se, entretanto, o que se quer é uma busca por ótimos globais, deve-se escolher uma razão de seleção mais genérica ou ampla, por exemplo (15,100) (Heitkoetter, 1999).

2.7.2 Programação evolutiva (EP)

Proposta por Fogel, Owens e Walsh em meados da década de 60. O livro seminal aqui é "*Artificial Intelligence Through Simulated Evolution*". Goldberg comenta que *"a rejeição deste trabalho pela comunidade da Inteligência Artificial, mais do que qualquer outro fator, foi a responsável pelo largo ceticismo quando comparado com os algoritmos genéticos ao final dos anos 60 e meados dos 70 (*Goldberg, 1989).

---

[1] A explicação para este procedimento, aparentemente contraditório, isto é: quando se está obtendo sucesso (taxa > 0,2) a variância deve ser aumentada, quando talvez o senso comum demandasse o contrário é simples. Se a taxa é maior do que 0,2 provavelmente ainda não se localizou o máximo global (se este existir) e portanto o ambiente evolutivo ainda "tateia" na busca dele. Conseqüentemente, deve-se aumentar a chance de sucesso nessa busca. Quando a taxa for menor que 0,2, ao contrário, espera-se que o máximo global (se existir) tenha sido encontrado, restando diminuir a taxa de mutação na busca do "ajuste fino".

Ainda que a proposta original tratasse de predição de comportamento de máquinas de estado finitos, o enfoque da programação evolutiva se adapta a qualquer estrutura de problema. Aqui, a representação sempre deriva do problema. Por exemplo, uma rede neural pode ser representada tal como é implementada, pois a mutação não exige uma codificação linear.

Cada indivíduo gera um único descendente através de mutação, e a seguir a (melhor) metade da população ascendente e a (melhor) metade da população descendente são reunidas para formar a nova geração. Usando a terminologia da estratégia evolutiva, esta implementação poderia ser nomeada como (μ+μ) (Bäck, 1993).

Em 1992, na sua tese de doutorado, chamada *Evolving Artificial Intelligence*, na Universidade da Califórnia em San Diego, Fogel apresentou o conceito de programação meta-evolucionária, na qual um vetor de variâncias substitui o valor padrão e exógeno da taxa de mutação, e aproxima este conceito da auto-adaptação descrita anteriormente para a estratégia evolutiva.

2.7.2.1 Algoritmo básico

O algoritmo básico segue as seguintes etapas:

Etapa 0: Escolhe-se uma população inicial de soluções de maneira randômica. O número de soluções é relevante para a velocidade da otimização.

Etapa 1: Cada solução gera uma nova população. Cada uma dessas soluções descendentes sofre mutação de acordo com uma distribuição de taxas de mutação

Etapa 2: Cada solução tem sua aptidão calculada. Os mais aptos são retidos como população de soluções. Não se exige que a população permaneça constante, ou que cada ascendente gere apenas um descendente.

A mutação é o único operador que atua na programação evolutiva. Não há recombinação.

2.7.3 Algoritmos Genéticos (GA)

Possivelmente o mais conhecido ramo da computação evolutiva, teve origem no trabalho de Holland, também nos anos 60. O livro seminal aqui é "*Adaptation in Natural and Artificial Systems*", (Holland, 1992b), cuja primeira edição é de 1975.

Ao contrário dos 2 esquemas vistos acima, (ES e EP), os algoritmos genéticos conceitualmente apresentam um escopo mais amplo do que a simples otimização. Eles são apresentados como *"um modelo de aprendizagem por máquina"* (Heitkoetter, 1999). Há uma explicação para este fato: originalmente, os algoritmos genéticos estavam muito fortemente ligados a modelos de aprendizado automático, como o demonstra a ênfase dada por Holland em (Holland, 1992a) aos chamados sistemas classificadores, que são um modelo de máquina de aprendizado usando algoritmos genéticos. Só mais tarde, basicamente a partir da publicação da tese de doutoramento de De Jong, e da edição do livro *Genetic Algorithms in Search, Optimization, and Machine Learning* (Goldberg, 1989), é que a idéia de otimização passou a ocupar o lugar central na teoria dos algoritmos genéticos. Em (Holland, 1992a), o autor introduz o assunto no âmbito da genética (pág. 32), economia (pág. 36), teoria de jogos (pág. 40), pesquisa, reconhecimento de padrões e inferência estatística (pág. 44), Controle e otimização de funções (pág. 54) e sistema nervoso central (pág. 58).

Dentro do ramo da computação evolutiva, muito se tem discutido a respeito do papel desempenhado pelos dois operadores básicos que são a mutação e a recombinação. Por exemplo, a estratégia EP estabelece que apenas a mutação deve ser usada. Pesquisadores têm discutido sobre a importância de ambos no âmbito dos algoritmos genéticos. Davis conta uma fábula, tendo como modelo a evolução natural, que pode esclarecer o papel dos dois operadores:

> *"A reprodução sexual é a empregada pelas criaturas mais complicadas do nosso planeta. Mais ainda, se comparada com a reprodução assexuada (envolvendo apenas 1 ascendente) o processo sexuado sai muito caro para gerar descendência. Espécies usando reprodução sexuada precisam ter mais de um tipo de indivíduo na espécie. Estes indivíduos devem diferir em importantes aspectos e eles tendem a gastar tempo e energia buscando seus parceiros antes da reprodução acontecer. Este custo adicional não ocorre na reprodução assexuada. Desde que a reprodução sexuada ganhou na arena da evolução, deve ser porque em alguns aspectos este custo adicional está muito bem recompensado.*

Uma resposta largamente aceita para esta questão é que a reprodução sexual permite a rápida combinação de novas características benéficas de uma maneira que não pode ser duplicada pela mutação. Consideremos por exemplo, 2 espécies de organismos ocupando o mesmo nicho ecológico e tendo a mesma estrutura cromossômica, exceto por um aspecto: os mutantes reproduzem-se com um único ascendente, aplicando-se depois o operador mutação. Já os cruzados reproduzem-se sexualmente com ambos os operadores (recombinação e mutação). Suponhamos que a taxa de mutação seja a mesma para as duas espécies e suponhamos mais ainda que há duas mutações, chamadas A e B, que aumentariam em muito a aptidão das duas espécies ao seu nicho. Finalmente suponhamos que a chance de A ou B ocorrerem seja de 1 em 1 bilhão.

Deixando as gerações passarem, vemos que nos mutantes, os indivíduos que têm A ou B começam aos poucos a dominar na sua população, mas ocorre um longo tempo antes que um indivíduo venha a ter ambas as mutações. Já com os cruzados existe uma diferença. À medida em que os indivíduos vão ganhando as mutações A ou B, cresce a chance de através de um casamento, ambas as mutações serem transmitidas à prole. O resultado final é que os cruzados eliminam os mutantes." (Davis, 1991).

Para promover as boas soluções, a técnica básica recebeu o nome de roda da roleta. A idéia é que todos os indivíduos de uma população sejam avaliados, e o resultado da avaliação seja usado como abertura angular em uma roleta de cassino. Em outras palavras, indivíduos aptos teriam um grande ângulo nesta roleta, enquanto indivíduos menos aptos teriam ângulos cada vez menores. Jogada a bola (que em termos computacionais sempre significará a geração de números pseudo-aleatórios), aqueles que tiverem maiores ângulos obviamente terão maior chance de serem escolhidos como ascendentes e é através deste mecanismo simples, que a aptidão média da população vai sendo incrementada.

Com o passar das gerações, percebe-se que as soluções "boas" começam a compartilhar partes comuns em seus cromossomos. Estas partes foram chamadas de esquemas, e o teorema fundamental dos algoritmos genéticos diz que esquemas que tiverem maior aptidão (o resultado da função de avaliação) do que a média da população tendem a crescer exponencialmente nas próximas gerações, enquanto que os esquemas que tiverem aptidões menores do que a média tendem a diminuir também expo-

nencialmente. Dizendo de outro modo, este teorema afirma que as soluções convergirão para um ponto de maior aptidão.

2.7.3.1 Teorema Fundamental dos Algoritmos Genéticos

Embora válido apenas para representações binárias de indivíduos, esta é talvez a primeira e mais importante formalização da garantia de funcionamento dos algoritmos genéticos.

Seja inicialmente, um alfabeto binário $V_1 = \{0,1\}$. Para efeito deste desenvolvimento necessita-se descrever o conceito de esquema, que vem a ser uma máscara, composta de três caracteres, quais sejam o 0, o 1 e um terceiro denominado "coringa" e representado por um *, que deve ser lido como "tanto faz". Portanto, para representar esquemas, necessita-se de um alfabeto trinário $V_2 = \{0,1,*\}$.

A finalidade do esquema é estabelecer cadeias e questionar se um indivíduo pertence ou não aquele esquema (cadeia).

> *Por exemplo: Seja o esquema **10. Pertencerão a este esquema os indivíduos 0010, 0110, 1010 e 1110. Não importa o que aparece nas duas primeiras posições, desde que nas duas últimas apareça 10. Esta é a leitura da cadeia **10.*

Se o esquema não possui nenhum *, apenas um indivíduo (se existente) pertencerá a esse esquema. Se possui um único *, dois indivíduos poderão pertencer a ele, e assim por diante. Posto assim, para um esquema de n asteriscos, o número de indivíduos que podem a ele pertencer é de 2^n.

Por outro lado, para indivíduos de tamanho k, podem existir 3^k esquemas diferentes,

> *Seja um conjunto de indivíduos de comprimento 2. Pela regra acima devem existir 9 esquemas, a saber: 00, 01, 10, 11, *0, *1, 0*, 1* e **.*

Em uma população de n indivíduos há sempre $n \times 2^m$ esquemas contidos na população, já que cada um deles é representativo de 2^m esquemas.

Alguns esquemas são mais genéricos que outros. Para qualificá-los surgem os conceitos de ordem e comprimento de esquema. A ordem de um esquema H, denotada o(H) é o número de posições fixas (ou seja, de uns e zeros) do esquema.

> *Por exemplo: se H_1=1 * * * 0, o(H_1) = 2, enquanto se H_2 = * * * * * *, o(H_2) = 0*

O comprimento de um esquema H, denotado por δ(H) é a distância entre o primeiro e o último zero ou um.

*Se H3 = 1 0 * * * 1 * *, δ(H3) = 6 - 1 = 5*

Com estes conceitos, pode-se analisar o efeito individual e coletivo dos operadores seleção, *crossover* e mutação sobre os esquemas contidos em uma população.

Analisando primeiro o efeito sobre a seleção (reprodução). Escreve-se m(H,t) como sendo o número m de exemplares de H presentes em uma população no instante t. Lembre-se de que há mais esquemas que indivíduos, vários esquemas estão representados por um único indivíduo, o que caracteriza o paralelismo de funcionamento do esquema global.

A probabilidade de um indivíduo *i* ser selecionado para reprodução é .

$$p_i = \frac{f_i}{\sum f_i}$$

onde f_i é a aptidão do indivíduo *i* e $\sum f_i$ é a somatória de todas as aptidões da população.

A partir deste conceito, pode-se estabelecer a quantidade esperada de esquemas na geração seguinte a uma dada, o que seria $m(H,t+1)$, e tem-se

$$m(H,t+1) = m(H,t).n.f(H)/\sum f_i$$

onde $m(H,t)$ é a quantidade de indivíduos que pertencem ao esquema H na geração t, n é a quantidade de indivíduos na população, e $f(H)$ é a aptidão média dos indivíduos que pertencem ao esquema H.

Mas, lembrando que a aptidão média de toda a população $\bar{f}$ pode ser escrita como

$$\bar{f} = \sum f \,/n$$

pode-se rescrever o processo de crescimento via reprodução de um esquema como segue:

$$m(H,t+1) = m(H,t)\frac{f(H)}{\bar{f}}$$

A leitura desta fórmula é: a quantidade de indivíduos que pertencem ao esquema H na próxima geração é igual a quantidade de indivíduos que pertencem ao esquema H na geração anterior, multiplicado pela aptidão média dos indivíduos que pertencem ao esquema H, dividido pela aptidão média da população.

Desta equação pode ser inferido o fato de que esquemas com aptidão maior do que a média da população, tendem a crescer ao longo do tempo, enquanto que esquemas com aptidão menor que a média tendem a diminuir.

Segue-se a análise do efeito do operador de *crossover* sobre a sobrevivência dos esquemas. Supondo um *crossover* de 1 ponto e um esquema de comprimento $\delta(H) = k$, sua probabilidade de se rompido é de

$$p_r = \delta(H)/k - 1$$

Um esquema sobrevive quando o corte da recombinação ocorre fora do seu comprimento definido e a probabilidade seria

$$p_s = 1 - \delta(H)/k - 1$$

Já que o própria recombinação ocorre com uma dada probabilidade, diga-se p_c, tem-se que a probabilidade de sobrevivência de um esquema H, a uma operação de *crossover* com probabilidade p_c de ocorrer é

$$p_s \geq 1 - p_c \cdot \frac{\delta(H)}{k-1}$$

Juntando as duas considerações (seleção e *crossover*), tem-se a seguinte probabilidade de sobrevivência de um esquema

$$m(H,t+1) \geq m(H,t) \cdot \frac{f(H}{\bar{f}} \left[1 - p_c \cdot \frac{\delta(H)}{k-1} \right]$$

Olhando esta fórmula, percebe-se que H cresce (ou diminui) dependendo de dois fatores conjugados: de que ela tenha aptidão maior (menor) que a média e pequeno (grande) comprimento.

Note-se que houve troca do sinal = pelo sinal ≥ uma vez que embora a recombinação possa ocorrer, nada impede que o resultado obtido seja exatamente igual ao que

havia antes da recombinação. Desta maneira, a probabilidade do esquema sobreviver pode ser maior ainda do que a calculada pela fórmula, daí o sinal de ≥.

Finalmente, vale considerar o efeito do operador de mutação. Um esquema sobrevive desde que todos os seus componentes não sejam modificados. Supondo uma mutação com probabilidade p_m, um alelo sobreviverá com probabilidade $(1-p_m)$. A probabilidade total de sobrevivência do esquema é obtida multiplicando-se $(1-p_m)$ por si mesmo tantas vezes quantas for a ordem do esquema. Ou seja,

$$p_{sm} = (1-p_m)^{o(H)}$$

Entretanto, como p_m sempre tem valores pequenos, a probabilidade de sobrevivência devida a mutação pode ser escrita como (Goldberg, 1989)

$$p_{sm} = 1 - o(H) \cdot p_m$$

Finalmente, juntando os 3 fatores que intervém na sobrevivência de um esquema, tem-se

$$m(H,t+1) \geq m(H,t) \cdot \frac{f(H}{\overline{f}} \left[1 - p_c \cdot \frac{\delta(H)}{k-1} - o(H) \cdot p_m \right]$$

E aqui chega-se à formulação matemática do que é conhecido como o teorema fundamental dos algoritmos genéticos: *esquemas curtos, de baixa ordem, de aptidão maior do que a média, surgem nas gerações subsequentes com número exponencialmente crescente.*

Tais esquemas com as características acima, recebem o nome de *building blocks,* ou blocos construtivos das soluções. A formulação aqui apresentada é a de Goldberg (1989). Em Mitchell (1997), há uma apresentação muito parecida, com apenas pequenos detalhes matemáticos ligeiramente distintos, mas equivalentes, por certo. Idem para Radcliffe (1997).

2.7.3.2 Algoritmo básico

Uma visão em alto nível do que seja um algoritmo genético.

Etapa 0. Inicializar a população de cromossomos (soluções).

Etapa 1. Avaliar cada cromossomo da população.

Etapa 2. Criar novos cromossomos a partir da população atual, aplicar mutação e recombinação, substituindo os ascendentes pelos descendentes.

Etapa 3: Se o critério de fim foi alcançado, deve-se terminar. Caso contrário, retorna-se à etapa 1.

2.7.3.3 Terminologia

A terminologia usada nos algoritmos genéticos é estreitamente relacionada ao universo genético natural, como pode-se ver na tabela a seguir, extraída de (Goldberg, 1989).

Tabela 1. Comparação entre a genética natural e os algoritmos genéticos

Mundo natural	Algoritmos genéticos
cromossomo	seqüência
gene	característica, caractere ou detetor
alelo	valor da característica
locus	posição da seqüência
genótipo	estrutura
fenótipo	conjunto de parâmetros, alternativa de solução, estrutura decodificada
epístase	não linearidade (ou influência de um gene sobre outros genes).

Esta explicação dos algoritmos genéticos é parcial e bastante simplificada. A literatura (Goldberg, 1989; Davis, 1991; Beasley, Bull e Martin, 1993a e 1993b; Heitkoetter, 1999 entre outros), registra os sucessivos melhoramentos que têm sido introduzidos na idéia básica no sentido de aumentar a generalidade e a robustez do método.

2.8 Idéias Comuns à CE

A inspiração básica para os posteriores desenvolvimentos aqui mostrados, é sempre o mundo natural. Os termos usados são tomados por empréstimo das ciências biológicas, usando-se algum tipo de analogia com o que já existe, embora como tenha dito Melanie Mitchel (1997), "*as entidades a que (os termos) se referem são muito mais simples do que aquelas do mundo real*". Por exemplo, quando se fala de recombinação no âmbito da CE, o fenômeno é uma extraordinária simplificação do mecanismo de troca de material genético que existe nos seres vivos da natureza.

2.8.1 Fitness Landscape

O conceito de *fitness landscape*, também conhecido como A*daptative surface* (Bäck, 1996) foi originalmente definido pelo biólogo Wright em 1931. É a representação do espaço de todos os possíveis genótipos, cada um com seu *fitness*. Supondo indivíduos de x cromossomos e com um número real k associado a cada cromossomo a *fitness landscape* (superfície de aptidão) é uma figura formada em um sistema de x+1 eixos. Nos x eixos representa-se o indivíduo através dos seus diversos valores em cada eixo, correspondendo aos valores de cada cromossomo que o compõe. Finalmente, no eixo que sobra representa-se o seu valor de *fitness*. Para efeito de clareza e exemplificação, seja um caso de indivíduos binários que tenham 2 cromossomos. Necessita-se portanto de 2 eixos para representar os indivíduos e em cada eixo estarão marcados os alelos possíveis para este cromossomo (eixo):

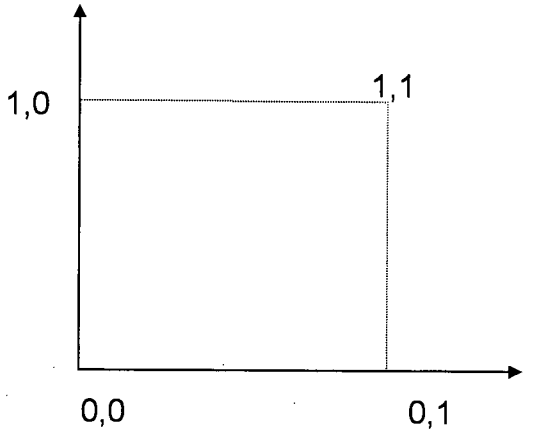


Fig. 3: Exemplo de *fitness landscape* para indivíduos com 2 cromossomos

Aqui estão representados os 4 indivíduos possíveis neste universo de cromossomos binários de comprimento 2: são eles os indivíduos 00, 01, 10 e 11. Supondo que esteja associado a cada um deles um determinado *fitness*, por hipótese, 0.9, 0.5, 0.4 e 0.0 e representado isso tridimensionalmente, ter-se-ia

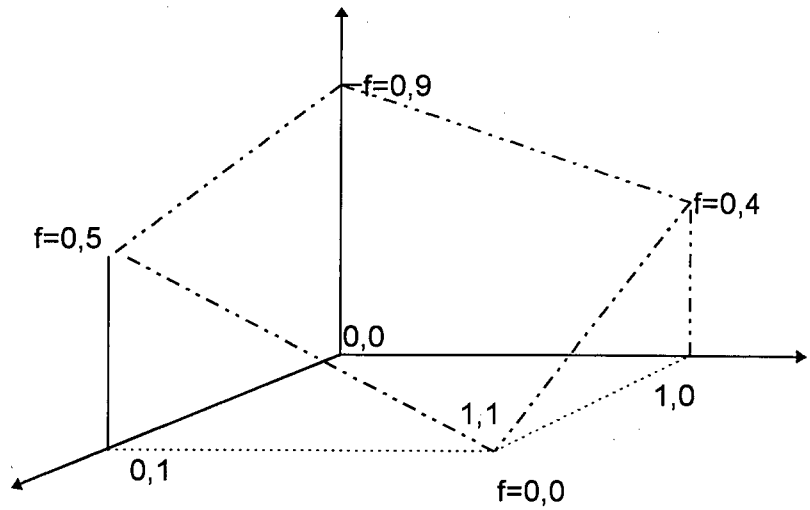


Fig. 4: Exemplo de *fittness landscape* para indivíduos com 3 cromossomos

À superfície formada pelos pontos (0,0,0.9); (0,1,0.5); (1,0,0.4) e (1,1,0) dá-se o nome de *fitness landscape* e ela representa todos os possíveis valores para a função objetivo que se está estudando. O nome *landscape* (paisagem) vem do fato de para problemas de maior dimensionalidade, ela é formada por montanhas, picos e vales entre outros acidentes geográficos. De acordo com a idéia original de Wright a evolução faz com que as populações se movam sobre essa superfície através de caminhos particulares. Adaptação (no sentido evolutivo) pode ser aqui entendido como o movimento ao redor de picos locais. Para encerrar a analogia, deve-se ressaltar que o *landscape* não é -- em casos gerais -- fixo, sendo dependente, principalmente no mundo real, da existência e da co-existência de outros indivíduos que iteragem entre si.

Esta idéia foi melhorada por Atmar em 1979 (Fogel, 1997b), quando este sugeriu que o conceito seria mais facilmente entendido se o plano fosse invertido, e os picos se transformassem em vales. Em vez da busca de alto valores de aptidão, o modelo agora buscaria minimização de erros. A diferença está na ação da gravidade sobre a paisagem. Rápidas descidas são seguidas por períodos de estagnação, principalmente se a paisagem é estática.

2.8.2 Hibridização

A técnica de hibridização, resulta na junção de uma boa maneira convencional de resolver um problema aos conceitos usuais da computação evolutiva. O resultado costuma ser melhor que o obtido com qualquer uma das duas técnicas utilizadas isola-

damente. Como diz Davis *"hibridizando um algoritmo genético com um algoritmo corrente, pode-se produzir um algoritmo que é melhor do que os dois anteriores"* (Davis, 1991). A hibridização agrega a representação de dados usual no domínio original, bem como as técnicas de otimização já porventura existentes. Isto permite a incorporação de heurísticas otimizadoras ao conjunto de operadores genéticos (recombinação e mutação) que passam portanto a ser dependentes do domínio. Nesse sentido, a computação evolutiva passa a ser muito mais uma filosofia de otimização do que um pacote de software pronto para usar.

Um exemplo de hibridização possível é quando o problema exige codificação com base em números reais e não em números binários. Alguns conceitos teriam que ser adaptados: por exemplo, a mutação não seria mais a troca simples de um bit, mas a geração de um novo real, possivelmente dentro de um intervalo dado. Já a recombinação de dois reais, poderia ser qualquer número compreendido entre eles, ou talvez a sua média.

Outra possibilidade de hibridização é quando o problema envolve algoritmos de ordem, como por exemplo, no célebre problema do caixeiro viajante (Um viajante precisa passar por um certo número de cidades, que estão ligadas por uma determinada malha viária. Associado a cada trecho do caminho, há uma função de penalização: custo, tempo, desgaste, entre outros. O objetivo do problema é resolver qual o trajeto que minimiza esta penalização). Ou no de colorir um grafo, que pode ser assim descrito: dado um grafo com certo peso em cada nodo e dadas *n* cores, o problema busca encontrar a mais alta soma de pesos de nodos pintados com alguma cor. Pode ser colorido qualquer nodo com qualquer cor, desde que nenhum par de nodos conectados tenham cor igual.

O algoritmo convencional para este problema (chamado de guloso) sugere a seguinte estratégia:

a) classificar o conjunto de nodos em ordem decrescente de peso

b) aplicar a cada nodo a primeira cor legítima que ele puder ter.

A crítica a esta solução é que ela sempre procede localmente. Não existe modo de encontrar uma solução global satisfatória, já que o problema é NP-completo. Para usar computação evolutiva, a questão é: como hibridizar este problema e a solução local? É necessário criar uma função de avaliação de um dado cromossomo, que agora representará seqüência de nodos de uma determinada cor. O operador de mutação pode ser o embaralhamento de uma sub-lista dentro do cromossomo. O operador de recombinação, pode permutar itens entre os dois ascendentes, associado a alguma função que impeça de gerar soluções inválidas. Em (Davis, 1991), onde este problema está

citado, há a descrição de um problema real (100 nodos e 3 cores) onde o algoritmo guloso obteve como resultado 9.590 pontos, o melhor resultado em 4000 gerações aleatórias obteve 9.610 pontos e um algoritmo genético hibridizado obteve 10.594 pontos.

2.8.3 Roda da roleta

Trata-se de um operador de seleção cujo propósito é obter maiores chances de reprodução no todo para aqueles membros da população de que tenham melhor aptidão. Segue um algoritmo semelhante a:

Etapa 0. Somar as aptidões de todos os membros da população. Chamar o resultado de aptidão total

Etapa 1. Gerar um número randômico n, entre 0 e a aptidão total

Etapa 2. Retornar o primeiro membro da população cuja aptidão, somada as aptidões anteriores seja maior ou igual a n.

Ainda que o procedimento de seleção seja randômico, cada ascendente tem uma chance de ser selecionado que é proporcional à sua aptidão. Tendendo ao limite, após um certo número de gerações, o algoritmo desprezará ascendentes com baixa aptidão e acabará valorizando aqueles com alta aptidão.

Seja um exemplo da técnica

Tabela 2 : Exemplo da técnica da Roda da Roleta

Número do cromossomo	1	2	3	4	5	6
Aptidão	11	7	1	21	3	16
Aptidão acumulada	11	18	19	40	43	59

Sejam agora a geração dos seguintes números aleatórios entre 0 e 59

Tabela 3 : Aplicação do exemplo de Roda da Roleta

Número aleatório gerado	36	8	27	51
Cromossomo selecionado	4	1	4	6

Neste exemplo, ao se gerar um aleatório entre 0 e 59, obteve-se 36. Percorrendo a tabela anterior, percebe-se corresponder este aleatório ao cromossomo nº 4 (já que 36 é maior que 19 e menor que 40). Identicamente para os números 8, 27 e 51.

2.8.4 Mutação de bit

É um operador evolutivo, conhecido como mutação. Trata-se da inversão aleatória de um determinado bit. Este operador ocorre com uma dada probabilidade, originalmente estabelecida em um valor baixo (por exemplo 0.01, significando 10 chances em 1000, ou 1%). Dada esta taxa T, diz-se que um bit tem probabilidade T de ser randomicamente substituído. Um exemplo:

Tabela 4 - Exemplo de uma mutação de bit

Cromossomo ascendente				Números randômicos				Novo bit	Cromossomo descendente			
1	0	1	0	.799	.109	.257	.289	-	1	0	1	0
1	1	0	0	.167	.890	.310	***.009***	1	1	1	0	**1**
0	0	1	1	.766	.673	***.003***	.345	1	0	0	1	1

Neste exemplo, dados os 3 cromossomos (denominados ascendentes na tabela acima), e dada uma taxa de mutação de 0.01, para cada bit dos cromossomos é gerado um aleatório. Sempre que o número gerado for superior a 0.01, a mutação não ocorre. Quando, ele for menor (casos do 4º bit do 2º cromossomo e do 3º bit do 3º cromossomo), o operador mutação gera um novo bit. Note-se que em 50% dos casos, o bit gerado é igual ao que havia antes (caso do 3º cromossomo), e nos outros 50% o bit é modificado. Em resumo, para uma taxa de mutação de 0,01, a taxa efetiva de mudança de bits é igual à metade deste valor.

2.8.5 Recombinação de 1 ponto

Ocorre quando partes de cromossomos ascendentes são invertidos após um ponto randomicamente selecionado criando 2 descendentes. Por exemplo, dado um ascendente

1	1	1	1	1	1

e outro ascendente

0	0	0	0	0	0

poderíamos ter um descendente

1	1	1	1	0	0

e outro descendente

0	0	0	0	1	1

se o corte tivesse se dado após o quarto bit nos ascendentes. Uma questão importante é que este operador produz descendentes que são radicalmente diferentes de

seus ascendentes. Outra, é que este operador não introduz nenhuma diferença nos bits das posições em que ambos os ascendentes tiverem o mesmo valor. Uma instância extrema ocorre quando ambos os ascendentes são completamente iguais. Também o serão os descendentes independente da ocorrência da recombinação de 1 posição.

2.8.6 Normalização linear

Técnica para melhorar o esquema de alocação de aptidão aos indivíduos da população. Ordena-se os cromossomos por seus valores decrescentes. Cria-se uma aptidão que começa em um valor constante e decresce linearmente. Este valor constante e a taxa de decremento são parâmetros da técnica

Exemplo

Tabela 5 : Exemplos de normalização linear						
Avaliação original	192	11	8	5	2	1
Aptidão é igual a avaliação	192	11	8	5	2	1
Normalização linear com início = 100, decremento = 3	100	97	94	91	88	85
Normalização linear com início = 100, decremento = 20	100	80	60	40	20	0

2.8.7 Janelamento

Técnica para aumentar a aptidão dos indivíduos que têm avaliação muito baixa. Acha-se o valor mínimo da população. Dá-se a cada cromossomo como aptidão o valor em que cada um excede este mínimo. Opcionalmente um valor pequeno acima deste valor mínimo pode ser dado aos piores cromossomos, já que na maneira original eles não teriam chance de reprodução.

Exemplo

Tabela 6 : Exemplos de Janelamento						
Avaliação original	192	11	8	5	2	1
Janelamento com mínimo = 0	191	10	7	4	1	0
Janelamento com mínimo = 5	191	10	7	5	5	5

2.8.8 Recombinação de 2 pontos

Nem sempre o operador de recombinação de 1 ponto consegue combinar certas características codificadas dentro dos cromossomos. Veja-se no exemplo a seguir dois cromossomos, nos quais os caracteres em negrito representam características importantes que devem ser preservadas (o que no linguajar de Holland, seria um *esquema*).

```
cromossomo 1: 0 1 0 1 1 0 0 1 0 1 1 0 1 1
cromossomo 2: 1 0 0 1 0 1 1 0 1 1 1 1 0 0
```

A recombinação de 1 ponto não consegue juntar estes dois esquemas em um único descendente, já que os pontos positivos do cromossomo 1 estão nas duas extremidades. Não importando onde o ponto seja escolhido, o cromossomo 1 vai se partir e seu esquema não passará à descendência.

Uma solução para este problema seria usar o operador de recombinação de 2 pontos. Ele funciona tal como o de 1 ponto, exceto que ele corta em 2 lugares aleatórios e o material é invertido entre os ascendentes nesses 2 pontos. Veja-se a seguir, como isso poderia ser feito no exemplo acima. Aqui, o sinal de = representa o local do corte.

```
Ascendente 1  : 0 1 0 1 = 1 0 0 1 0 1 = 1 0 1 1
Ascendente 2  : 1 0 0 1 = 0 1 1 0 1 1 = 1 1 0 0

Descendente 1: 0 1 0 1   0 1 1 0 1 1   1 0 1 1
Descendente 2: 1 0 0 1   1 0 0 1 0 1   1 1 0 0
```

2.8.9 Recombinação uniforme.

Para cromossomos e esquemas mais sofisticados, foi desenvolvida uma recombinação que tem a possibilidade de cortar um cromossomo em até n pontos, onde n é o comprimento do mesmo. Ele funciona como segue: 2 ascendentes são selecionados e 2 descendentes são produzidos. Para cada posição de bit nos 2 descendentes, decide-se randomicamente qual ascendente contribuirá com seu valor de bit para qual descendente.

Veja-se no exemplo

```
Ascendente 1    : 0 0 0 1 0 1 1
Ascendente 2    : 1 1 0 1 1 0 1

Máscara : 1 1 0 1 0 0 1 (funciona como selecionador do ascendente)

Descendente 1  : 0 0 0 1 1 0 1
Descendente 2  : 1 1 0 1 0 1 1
```

A máscara indica qual ascendente cederá seu bit para o descendente 1. A negação da máscara cria o descendente 2.

2.9 Comparação

A tabela a seguir, retirada de (Bäck, 1993 - pág. 19 e Bäck 1996, pág 132) retrata bem as diferenças conceituais sobre as três abordagens:

Tabela 7 : Comparação entre ES, EP e GA

Característica	Estratégia Evolutiva	Programação Evolutiva	Algoritmos Genéticos
Representação	números reais	números reais	*strings* binários
Auto-adaptação	desvio padrão e covariância	Variância	não há
Aptidão	valor da função objetivo	valor da função objetivo após normalização	valor da função objetivo após normalização
Mutação	operador principal	operador único	operador secundário
Recombinação	variantes distintas, importante para a auto-adaptação	não há	operador principal
Seleção	determinística e extintiva	probabilística e extintiva	probabilística e preservativa
Restrições	Restrições, inequações e arbitrário	não há	limites simples, providos pelo mecanismo de codificação
Teoria	Taxa de convergência para casos especiais, (1+1)-ES, (1+λ)-ES, (1,λ)-ES, convergência global para (μ+λ)-ES	Taxa de convergência para casos especiais, (1+1)-EP, convergência global para (1+1)-EP	teoria do processamento de esquemas, convergência global para a versão elitista

Segundo Heitkoetter (1999), há duas diferenças importantes entre EPs e GAs. Primeiro na EP não há restrições de representação, enquanto em GA, os cromossomos devem ser seqüências de bits representativos da solução do problema. Segundo, na EP o operador mutação modifica aspectos da solução de acordo com uma distribuição estatística, que modifica a taxa de mutação em função do grau de proximidade do ponto solução em relação a um ponto de máximo, enquanto para os GAs, a taxa de mutação permanece constante.

Ainda segundo o mesmo autor, as diferenças entre ESs e EPs são: na seleção, EP usa uma seleção estocástica através de uma competição, que elimina os perdedores da população. Em contraste, ES usa uma seleção determinística na qual as piores solu-

ções são eliminadas da população diretamente a partir da avaliação de aptidão. A outra diferença se dá na recombinação. EP é uma abstração da evolução das espécies, e como não há recombinação entre espécies diferentes, EP não possui este operador. Já ES implementa a evolução no nível de indivíduo, e portanto são aceitas formas de recombinação.

Quanto à seleção, tanto em ES como em EP, alguns indivíduos são definitivamente excluídos da seleção, pelo que Bäck usa o termo extintivo na tabela acima. Já em GA, pelo menos na sua versão canônica, cada indivíduo tem uma probabilidade maior que zero de ser selecionado, pelo que a tabela descreve a seleção em GA como preservativa (Bäck, 1993).

Uma diferenciação interessante pode ser estabelecida entre duas classes de implementações: algoritmos genéticos e programação genética usualmente aplicam primeiro o operador de seleção para obter boas soluções e então aplicam a recombinação e a mutação. Em contrapartida, estratégia evolutiva e programação evolutiva invertem a ordem, aplicando primeiro recombinação e mutação, de maneira a gerar um conjunto de soluções e então usam o operador de seleção para obter um subconjunto ótimo (Deb, 1997).

3. Problemas Inversos Abordados

> An *inverse problem* is to determine: "...unknown causes based on observation of their effects. This is in contrast to the corresponding direct problem, whose solution involves finding effects based on a complete description of their causes."
>
> *O.M. Alifanov*

3.1 Introdução

Qualquer fenômeno físico pode ser modelado descrevendo-se 3 conjuntos, que serão aqui chamados de E, T e S. O conjunto E, corresponde aos dados de entrada do modelo. O conjunto S é formado pelos valores de saída, que são obtidos a partir dos valores de E fornecidos ao modelo. T é uma transformação que habilita a obtenção de S a partir de E. Posto dessa maneira, resolver um problema físico direto é *"obter uma descrição precisa de T"* (Sabatier, 1985). Mediante essa descrição precisa, para qualquer conjunto de valores $e \in E$, é possível obter seus equivalentes valores $s \in S$.

Muitas vezes, em problemas do mundo real, o que se tem é o conjunto de valores $s \in S$, e a partir destes necessitam-se os correspondentes em E. Se a transformação T fosse bijetora (o que garantiria existência e unicidade de soluções) com T e T^{-1} contínuas e preferencialmente deriváveis (o que garantiria a estabilidade dos problemas direto e inverso), ferramentas convencionais da análise matemática poderiam resolver a questão. Entretanto, este caso "bem posto" raramente é o caso físico (Sabatier, 1985).

Quando então, não se tem a possibilidade de obter T^{-1} simplesmente a partir de T, outras abordagens de solução de problemas inversos têm que ser utilizadas, e neste caso a CE aparece como candidata promissora. Note-se que ao se abrir mão (até pela impossibilidade) da obtenção de T^{-1}, fatalmente precisa-se aceitar algum grau de degeneração dos resultados. Devido a essa característica, a teoria de problemas inversos foi chamada de "teoria do diagnóstico" (Sabatier, 1985), na qual decisões são tomadas em

ambientes com graus variáveis de incerteza. A solução de um problema inverso pode ser aceita quando, mesmo sem chegar a um resultado único e definitivo, ela levar a

- respostas convergentes ao resultado definitivo e correto, ou
- um conjunto maior de informações referentes a esse resultado, ou
- grau maior de certeza quanto a decisões tomadas sobre um problema aplicado, ou
- qualquer combinação das 3 acima, de maneira otimizada, isto é, garantindo um adequado compromisso entre custo computacional e resultados obtidos.

Percebe-se nessa rápida descrição de problemas inversos, como há uma vocação da CE para tratá-los. De fato, as premissas de ambos os campos são bastante próximas, a saber:

- aceitação de resultados aproximados, o que poderia ser chamado de "aceitação do menos pior".
- característica probabilística dos resultados obtidos.
- complexidade do ambiente sob estudo.
- inexistência, ou impossibilidade prática de uma solução melhor ou mais facilmente encontrável.

3.2 Regularização

C de Mol (1992) descreve uma "doença" dos problemas inversos a que ele chamou de *ill-posed,* que se pode traduzir por *mal-postos*. Para entender o conceito se faz necessário descrever o que são problemas bem-postos, por oposição àqueles.

Problemas bem-postos (well-posed), propriamente postos ou corretos possuem completamente os seguintes 3 requisitos: suas soluções possuem existência, unicidade e continuidade com relação aos dados. Basta que um problema deixe de ter um requisito acima para se transformar em mal-posto. A característica principal de problemas *ill-posed* é uma instabilidade que surge ao se analizarem dados experimentais. Acrescente-se o fato de que em dados experimentais sempre está presente algum nível de ruído, o que ajuda a ressaltar a característica de instabilidade do problema.

Os problemas bem-postos foram inicialmente enfocados por Hadamard em seus trabalhos no início deste século. Já os mal-postos foram considerados por este como *meras curiosidades* (C de Mol, 1992) fora do alcance da física-matemática. Nos anos 50,

a partir de trabalhos com equações diferenciais, começou a surgir o enfoque da regularização, no qual problemas ill-posed são convertidos em well-posed pela restrição *a priori* da classe de soluções graças a restrições apropriadas que são introduzidas com o nome de alavancas estabilizadoras. O efeito destas restrições é expurgar os componentes da solução generalizada que são muito sensíveis aos erros nos dados. Por exemplo, um estabilizador deste tipo é a chamada compacidade, sugerida por Tikhonov em 1943. Neste, restringe-se o espaço E a um subconjunto H e com isto, se a transformada T^{-1} existe, ela é contínua.

Usando a nomenclatura definida no início deste capítulo define-se um problema inverso como a solução da equação $L.f = g$, onde f pertence ao espaço E chamado espaço de soluções, g pertence ao espaço S denominado espaço de dados e $L : E \rightarrow S$ é a transformação que leva um objeto a sua imagem. Define-se o operador $L^* : S \rightarrow E$ que mapeia os dados g em suas correspondentes soluções f*, como a inversa generalizada do operador L. Ela é definida no domínio de L e coincide com a inversa usual L^{-1} quando esta existe.

Um algoritmo de regularização {Rμ} dependente de um parâmetro positivo μ provê limites aproximados para a inversa generalizada , ou seja

$$\lim_{\mu \to 0^+} R_\mu g = L^* g$$

Um método de regularização é um algoritmo de regularização completado pela escolha de um parâmetro $\mu = \mu(\varepsilon, g^\varepsilon)$ que assegure que para qualquer dado ruidoso $g^\varepsilon \in S$ na ε-vizinhança dos dados exatos (ou seja onde $\|g - g^\varepsilon\| \leq \varepsilon$) tem-se

$$\lim_{\substack{\varepsilon \to 0 \\ g^\varepsilon \to g}} R_{\mu(\varepsilon, g^\varepsilon)} g^\varepsilon = L^* g$$

O parâmetro μ, chamado parâmetro de regularização é escolhido de tal maneira que

$$\lim_{\substack{\varepsilon \to 0 \\ g^\varepsilon \to g}} \mu(\varepsilon, g^\varepsilon) = 0$$

Se μ depende apenas da precisão ε dos dados e não dos dados g^ε propriamente ditos, diz-se que há uma escolha *a priori* para μ. Se μ depende de g^ε diz-se que a escolha é a posteriori.

Termina-se reconhecendo que a instabilidade prevista na teoria se fez presente sobre todos os dados dos problemas estudados. Como se verá a seguir, pequenas variações na entrada causaram variações várias ordens de grandeza maiores na saída. Resta considerar a citação de Lanczos (1961) que diz *"a falta de informação não pode ser remediada por nenhum artifício matemático".*

3.3 Estudo de Casos: Tratamento Clássico

O objetivo desta seção é apresentar os três problemas inversos que serão estudados nesta Tese. Será inicialmente mostrado o esquema de resolução clássico que vai servir de comparação com o tratamento evolutivo.

3.3.1 Reconstrução das distribuições de condutividade geoelétrica

Este problema conforme descrito em Ramos e Campos Velho, (1996), busca *reconstituir a distribuição de condutividade geoelétrica da terra usando um conjunto incompleto e ruidoso de medidas do campo eletromagnético na superfície terrestre.*

O problema direto é definido como a obtenção de valores do campo eletromagnético da terra, a partir de medidas de condutividade elétrica colhidas no campo ou simuladas. Um plano bi-dimensional que corte a superfície terrestre é estabelecido e nele prismas com diferentes medidas de condutividade são individualizados de modo que se possa obter a medida do campo eletromagnético pelas equações de Maxwell.

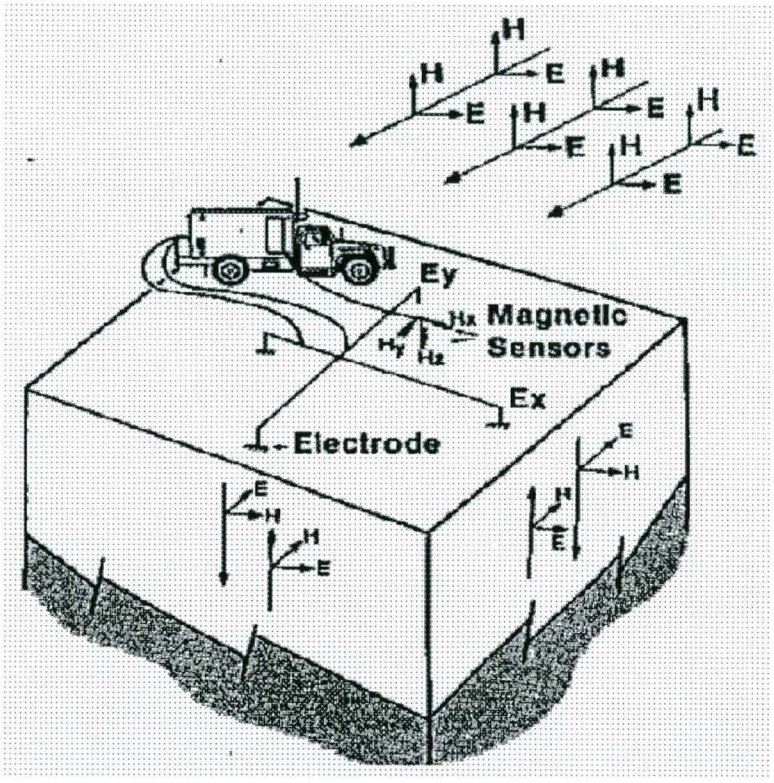


Fig. 5: Uma representação esquemática do problema de inversão magnetotelúrica

Já o problema inverso assume que estão disponíveis diversas medidas do campo eletromagnético obtidas na superfície da terra em uma série de pontos e para diversas freqüências. A formulação matemática do problema inverso tratado em Ramos e Campos Velho (1996) com a metodologia lá descrita pode ser dada como

$$\min J(\gamma, p), \quad l_q \le p_q \le u_q, \quad q = 1, \ldots, Q$$
$$J(\gamma, p) = R(p) - \gamma_0 S_0(p) / S_{max} + \gamma_1 S_1(p) / S_{max}$$

onde

p_q é o vetor de condutividades a ser determinado pela análise inversa,

l_q e u_q são valores previamente escolhidos para fazer a inversão existir dentro de limites físicos conhecidos,

γ_0 e γ_1 são parâmetros de regularização positivos,

S_0 e S_1 são funções de regularização,

S_{max} é uma constante de regularização.

Já o desvio entre o modelo e os dados, R(p) é definido por

$$R(p) = \sum_{j=1}^{N_y} \sum_{m=1}^{M} \left[\Phi_{j,m}^{E} - \Phi_{j,k^*,m}(p) \right]^2$$

onde

$\Phi^{E}_{j,m}$ é a medida do campo eletromagnético nas posições j = 1,2,..., N_y, e nas freqüências m = 1,2,...,M na superfície terrestre,

$\Phi_{j,k^*,m}$ é a medida do campo eletromagnético no nível k*,

p é o vetor de condutividades.

Foi introduzida uma regularização baseada na medida da entropia S_1 do vetor das primeiras diferenças de p, chamado método da mínima entropia de primeira ordem, ou MINENT-1

Também usou-se o método de entropia máxima o qual usa a entropia de ordem zero de p, ou MAXENT-0. Os termos de regularização das equações acima podem ser dados por

$$S_\alpha(p) = -\sum_{q=1}^{Q} s_q \log(s_q)$$

$$s_q = r_q / \sum_{q=1}^{Q} r_q \quad \text{onde } r_q = \begin{cases} p_q, & se\ \alpha = 0 \\ p_q - p_{q-1} & se\ \alpha = 1 \end{cases}$$

As funções S_0 e S_1 atingem seus pontos máximos globais quando todos os r_q são iguais, o que corresponde a uma distribuição uniforme com o valor de S_{max} = log Q. Por outro lado, o mais baixo nível de entropia S_{min} = 0, é atingido quando todos os elementos, exceto um, são iguais a zero.[2]

---

[2]Uma analogia deste fato pode ser obtida ao se analisar o comportamento de um dado. Ao se ter um dado viciado que sempre apresenta o mesmo resultado, (digamos que as probabilidades de cada uma das 6 faces sejam dadas por {1,0,0,0,0,0}) diz-se que a entropia é baixa e que há um alto conhecimento para inferências futuras do comportamento do dado. Por outro lado, com um dado não viciado, cujas probabilidades por face são dadas por {1/6, 1/6, 1/6, 1/6, 1/6, 1/6}, diz-se que a entropia é alta e consequentemente

Enquanto o método MAXENT-0 busca uma regularidade global, o MINENT-1 busca regiões localmente regulares separadas por descontinuidades abruptas. Como dizem Ramos e Campos Velho (1996), "*qualquer reconstituição compartilhando estas características tem um alto nível de informação e portanto um conteúdo de baixa entropia. Esperamos muitas propriedades geofísicas interessantes para comportar-se de maneira similar*".

O valor dos parâmetros de regularização, foi estimado em função de experimentação numérica. Assumindo que os dados de entrada possuem erro com uma distribuição gaussiana com desvio padrão $\sigma_{j,m}$, foram buscados os valores que aproximam a estatística

$$\sum_{j,m}(\Phi^{E}_{j,m} - \Phi_{j,k^*,m})^2 / \sigma^2_{j,m}$$

ao seu valor esperado, a saber o número total de observações (N_y.M).

No trabalho descrito em Ramos e Campos Velho (1996), usou-se uma biblioteca de otimização denominada *NAG Fortran Library*. De fato, na pág. 5 do citado texto consta "*na ausência de uma solução explícita para um problema não linear definido por estas equações uma pesquisa interativa é feita usando uma rotina de minimização da biblioteca FORTRAN NAG*".

É exatamente neste ponto que a computação evolutiva vai ser testada. Pois, no estudo a ser desenvolvido vai-se comparar o desempenho e os resultados alcançados pela computação evolutiva *vis-a-vis* aqueles obtidos pela rotina NAG.

Os resultados numéricos obtidos em (Ramos e Campos Velho, 1996) são o resultado da aplicação do método MINENT-1 a uma estrutura consistindo de um prisma condutivo Ω^c e um prisma resistivo Ω^r, ambos inseridos no semi-plano Ω^+ com $\zeta^c / \zeta^r = 10$ e $\zeta^r / \zeta^+ = 0.1$. O modelo foi discretizado em um conjunto de 13 x 11 blocos, sendo que dos 143, 88 estão inseridos no semi-plano condutivo, isto é, abaixo da superfície terrestre. Os dados foram gerados pelas equações diretas para 11 pontos na superfície terrestre e em 20 freqüências logaritmicamente espaçadas variando no intervalo entre 0.0001 e 0.01 Hz. Foi adicionado um ruído gaussiano com taxa de 1%. Os resultados foram computados para 4 casos:

---

também o é a ignorância em relação ao futuro. Este fenômeno foi chamado por Shannon de *missing information.*

- Sem regularização ($\gamma_0 = 0$ e $\gamma_1 = 0$)
- Regularização MINENT-1 ($\gamma_0 = 0$ e $\gamma_1 > 0$)
- Regularização MAXENT-0 ($\gamma_0 > 0$ e $\gamma_1 = 0$)
- Regularização híbrida ($\gamma_0 > 0$ e $\gamma_1 > 0$)

O trabalho, baseado nos resultados numéricos lá apresentados, conclui pela adequação do método MINENT-1, "*ainda que computacionalmente caro*" e sugere mais pesquisas buscando a diminuição do custo computacional da otimização e a conveniente expansão do modelo para conseguir operar sob 3 dimensões. No que concerne à primeira sugestão, o uso de CE é uma tentativa — o que será abordado e estudado na tese.

3.3.2 Transmissão de Calor

Este problema foi estudado no âmbito do INPE e UNIVAP e busca estudar a composição de uma placa similar a um tabuleiro de xadrez, formada por 2 materiais com condutividades térmicas distintas. A quantidade de "casas" de cada material é arbitrária. O cálculo da condutividade térmica equivalente do conjunto passa pela análise da placa que é constituída segundo uma determinada construção regular. O modo direto de cálculo é obtido pela descrição da distribuição dos componentes na placa. Pelo conhecimento das condutividades dos componentes individuais é possível calcular-se a condutividade equivalente. Já o problema inverso pode ser formulado como "dada uma condutividade da placa, inferir quais são e onde devem estar seus componentes cujas condutividades individuais são conhecidas. Novamente o cálculo direto é obtido na modalidade de caixa preta, a partir de programação em Fortran. Este programa usa como minimizador um algoritmo de *simulated annealing*.

3.3.3 Difusão de calor em materiais compostos

Este problema conforme descrito em Ramos e Giovannini (1994) busca obter a reconstituição tomográfica de defeitos em sólidos uni, bi e tri-dimensionais feitos de materiais compostos.

Os materiais compostos têm como característica maximizar o compromisso entre massa volumétrica e propriedades mecânicas. Segundo os autores, "*atualmente o emprego de materiais compostos generalizou-se rapidamente em domínios tão diversos*

quanto o nuclear, a aeronáutica, o espaço e a micro-eletrônica" (Ramos e Giovannini, 1994).

O problema direto considera uma estrutura plana composta, de dimensões $L_x \times L_y \times L_z$ constituída de *I* camadas paralelas de espessura $\delta x_i = x_{i+1} - x_i$, na qual são estipulados pequenos prismas, formando uma grade de discretização. Dados, a condutividade térmica do material (medida em W m^{-1} K^{-1}), o coeficiente de difusão térmica (medido em m^2 s^{-1}), a espessura das camadas (medida em m), a condutância térmica de contato (medida em W m^{-2} K^{-1}) e finalmente o pulso térmico aplicado ao material (duração e intensidade), o modelo direto exprime a temperatura em cada um dos pontos da grade.

O problema inverso, busca determinar o vetor solução p, através de mínimos quadrados que minimizam o valor quadrático entre o modelo direto e os dados experimentais (Ramos e Giovannini, 1994), buscando determinar características geométricas e físicas de defeitos situados no interior da peça, a partir de temperaturas tomadas em sua superfície.

Em termos algébricos, o problema inverso pode ser posto como

$$min\{J_\gamma(p)\}_{p\in P} \qquad l_q \le p_q \le u_q \qquad q = 1,\ldots,Q$$

onde

$$J_\gamma(p) = R(p) - \gamma S(p)$$

e

$$R(p) = \sum_{k=1}^{K} \sum_{l=1}^{L} \sum_{m=1}^{M} \left[Y_{kl}^m - T_{1kl}^m(p) \right]^2 .$$

onde J_γ é a função a minimizar

R(p) é a soma das diferenças quadráticas das temperaturas entre o modelo e os dados experimentais

S(p) é uma função de regularização, e

$\gamma > 0$ é o parâmetro de regularização

A formulação original do problema em Ramos e Giovannini (1994), prevê o uso de um algoritmo iterativo de otimização não linear quasi newtoniano da biblioteca NAG.

Na conclusão, Ramos e Giovannini escrevem "as possibilidades do método de inversão aqui proposto, só são limitados pelo tempo de CPU[3] necessário à convergência do algoritmo de inversão, que varia na razão do quadrado do número de parâmetros desconhecidos do problema e também em função da qualidade dos dados experimentais", e logo adiante, arrematam "falta fazer progressos no nível de otimização dos tempos de cálculo para que esta técnica de inversão seja transportável e utilizável no meio industrial" (Ramos e Giovannini, 1994).

É nesta última frase do trabalho, que se consubstancia o objetivo da tese: buscar métodos que permitam acelerar o processo de convergência necessário à implementação do acima referido problema inverso.

3.4 Estudo de Casos: Tratamento Evolutivo

Neste tópico abordam-se as abordagens evolutivas para os problemas acima descritos. Ressalte-se que os modelos de solução direta vistos continuam a ser utilizados na abordagem evolutiva. A ênfase aqui é no otimizador, que é o engengo evolutivo.

3.4.1 Problema da Reconstrução das Distribuições de Condutividades Geolétricas

Para poder abordar o problema inverso, deve-se obter a solução do problema direto. No trabalho original de Ramos e Campos Velho (1996), descreve-se o modelo matemático baseado nas Leis de Maxwell que permite sua solução. Aqui, o modelo direto é usado na modalidade de "caixa preta".

Cada indivíduo na população é representado por uma matriz de 7 x 10 números reais cada um deles associado a uma fatia retangular (um prisma cortado por um semi-plano) de condutividade desconhecida. Suas condutividades multiplicam a expressão $4\pi \times \omega \times 10^{-10}$ e são expressas em mhos/m. As dimensões dos prismas são Δy = 10 km e Δz variando entre 1 e 10 km. A função objetivo a ser maximizada é

$$f = \frac{1}{(1+k)^{\varepsilon}}$$

onde f é o fitness de cada indivíduo, K é uma constante que atua como um parâmetro que permite ajustar a pressão evolutiva sendo igual a 0.01 no presente caso e ε é o erro magnético, dado por

---

[3] CPU = Central Processing Unit

$$\varepsilon = \sum_{i=1}^{440} |H_{pi} - H_{ci}|$$

Na equação acima, H_p são as componentes do campo magnético gerado pelo indivíduo padrão (ou obtidas no campo) e H_c são as 440 componentes calculadas pelo engenho evolutivo. Por simplicidade de apresentação, mas sem perda de generalidade, apenas o campo magnético é considerado nesta tese. O número 440 resulta no fato de que estão sendo medidos 11 pontos da superfície terrestre em 20 freqüências (variando entre 0.0001 e 0.01 Hz) com componente real e imaginária para cada ponto.

Para um problema real, ε seria o único erro possível de ser conhecido. Aqui, entretanto, por se tratarem de dados sintéticos, é sabido o indivíduo original que se busca reconstituir com a técnica. Assim, embora a pesquisa evolutiva seja guiada pelo erro ε, é possível definir um segundo erro, denominado erro condutivo (E), dado por

$$E = \sum_{j=1}^{70} |C_{pj} - C_{cj}|$$

ou seja, E (erro condutivo) é o somatório em valor absoluto das 70 diferenças entre as condutividades padrão (C_p) e calculadas (C_c). Uma justificativa para apresentar os 2 erros está no fato de que, em problemas inversos, nem sempre a minimização de um erro implica a minimização do outro. Esta é uma das características dessa classe de problemas que dificultam a sua solução.

Graficamente, o problema pode ser representado no fluxo abaixo.

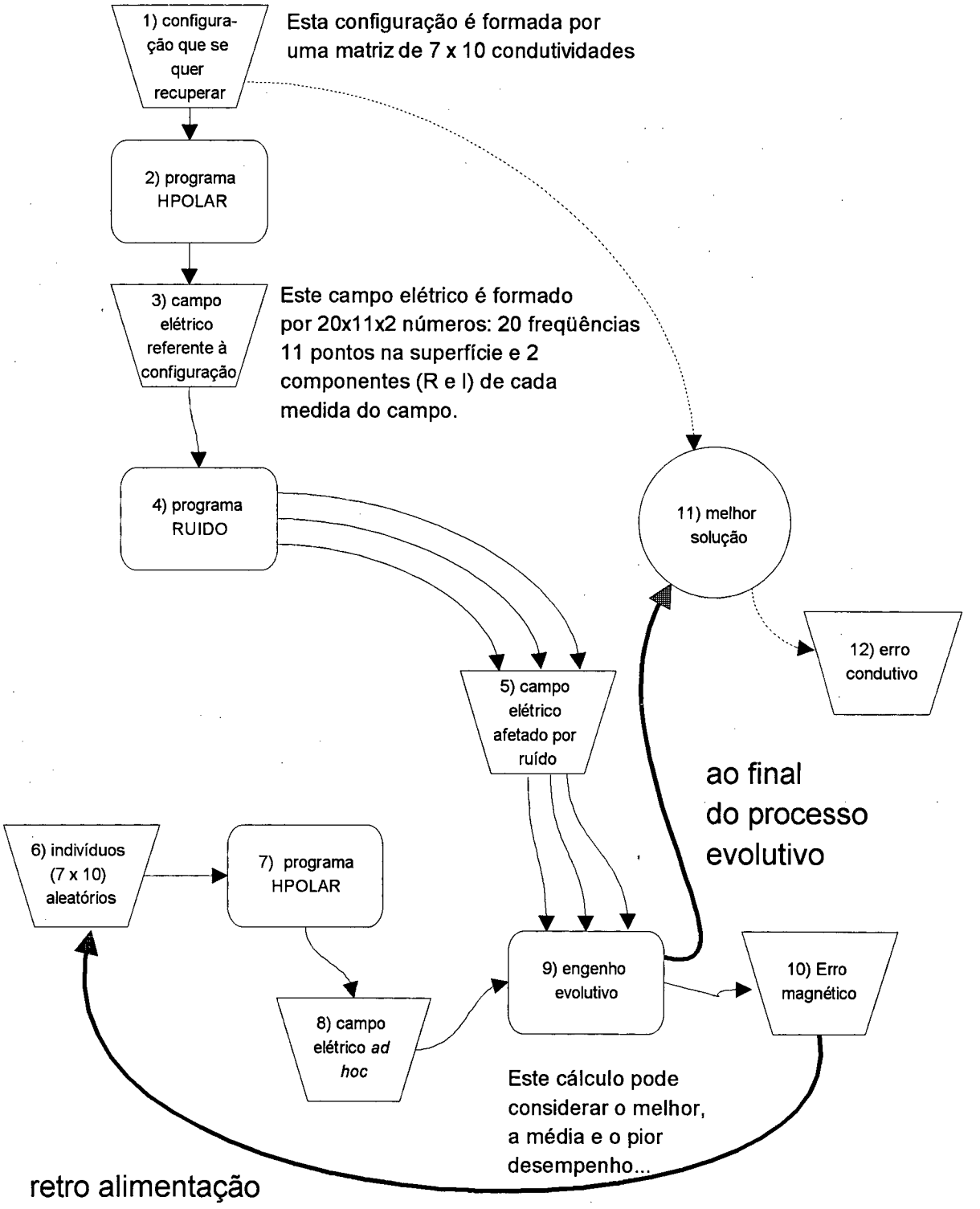


Fig. 6: Representação gráfica do problema de inversão magnetotelúrica

A configuração inicial representada na forma de uma matriz de condutividades (número 1) é o padrão que se busca reconstituir. Após processado pelo programa HPOLAR (2), obtém-se um campo elétrico (3). Este ciclo existe uma vez que os dados são sintéticos. Quando o modelo vier a ser utilizado com dados reais de campo, esta primeira fase não existirá, iniciando-se o ciclo diretamente em (5).

Para simular dados reais, a saída (3) sofre ainda a atuação do programa RUÍDO (4) que adiciona um ruído gaussiano gerando os dados ruidosos (5).

De outra parte, em (6), indivíduos (condutividades) são gerados inicialmente de maneira aleatória e depois a partir das regras evolutivas, obtendo-se candidatos a solução do problema.

Cada um deles, em (7) sofre a ação do programa HPOLAR, que é o mesmo de (2) e que gera campos elétricos específicos de cada candidato, representado na figura por (8).

O engenho evolutivo (9), compara este campo com 3 medidas do padrão que se busca reconstituir, sendo as 3 devidamente alteradas por ruído. O ruído é aplicado por meio de uma distribuição normal, cuja média é sempre igual a zero, e desvio padrão é uma percentagem da medida. O fato da média ser zero implica que poderá haver erros positivos e negativos. Obtém-se na comparação o erro magnético. Pode-se optar pela menor diferença (o melhor indivíduo), pela média delas, ou pela maior (o pior indivíduo). Seja como for, em (10) tem-se um único número que dá o desempenho do indivíduo gerado em (6).

O ciclo 6, 7, 8, 9 e 10 se repete para cada indivíduo e pelo número de gerações determinado. Ao final, em 11, obtém-se o melhor resultado daquela rodada.

Já que os dados são sintéticos e portanto o resultado (1) é conhecido, pode-se ainda calcular o segundo erro (condutivo), em 12. Este valor serve para qualificar a solução encontrada, mas deve-se ressaltar que ele não serve como guia durante o processo de busca. Quem exerce este papel isoladamente é o erro magnético.

3.4.1.1 Configurações Estudadas

Como visto, um resultado qualquer é expresso na forma de uma matriz de 70 números reais (7 linhas por 10 colunas). A seguir apresentam-se todas as configurações que foram estudadas nesta etapa do problema. Os valores estão expressos em mhos/m como coeficientes de 10^{-10}.

3.4.1.1.1 Configuração 1

Esta configuração pressupõe a existência de 2 blocos de material distintos dentro da rocha. Ambos são regulares e de mesmo tamanho (4 prismas cada um). O primeiro tem condutividade 10 vezes menor do que a rocha e o segundo a tem 10 vezes maior. Comparando ambos os materiais tem-se uma diferença relativa de 100:1. Numericamente eis o padrão de condutividades utilizado:

```
10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0
10.0  10.0   1.0   1.0  10.0 100.0 100.0  10.0  10.0  10.0
10.0  10.0   1.0   1.0  10.0 100.0 100.0  10.0  10.0  10.0
10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0
10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0
10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0
10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0
```

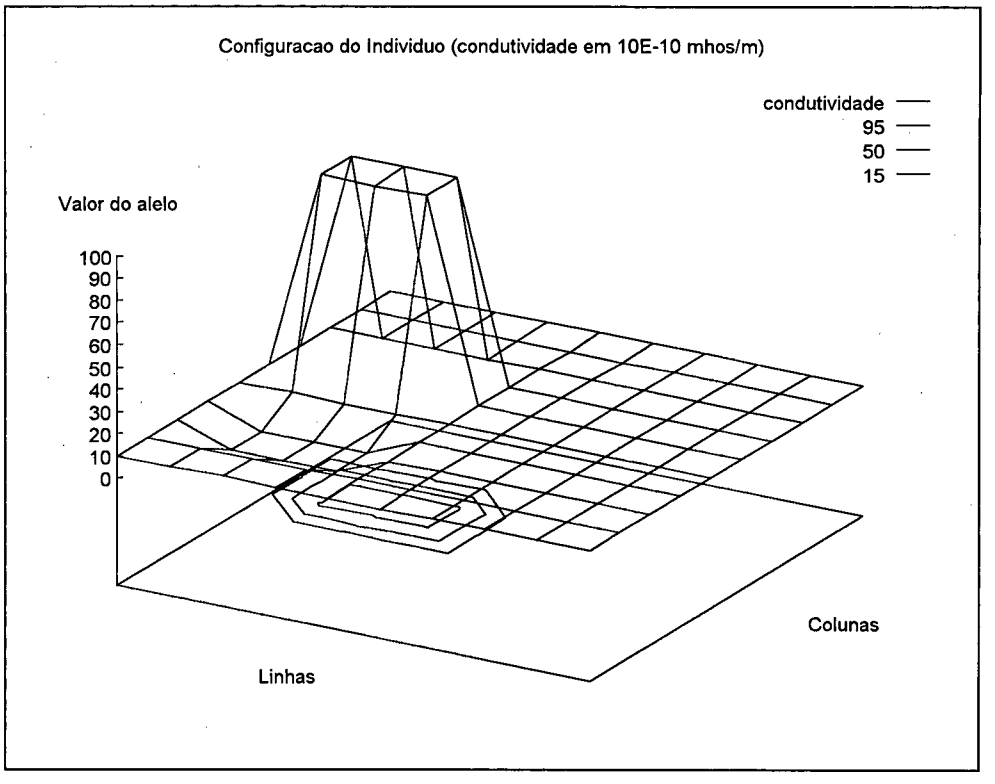


Fig. 7 : Representação da configuração 1 usada na inversão magnetotelúrica

3.4.1.1.2 Configuração 2

A segunda configuração apresenta um único bloco de material distinto dentro da rocha. Trata-se de um bloco de 10 prismas, retangular e cuja condutividade é 10 vezes maior que a da rocha circundante. Sua representação numérica:

```
10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0
10.0  10.0 100.0 100.0 100.0 100.0 100.0  10.0  10.0  10.0
10.0  10.0 100.0 100.0 100.0 100.0 100.0  10.0  10.0  10.0
10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0
10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0
10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0
10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0
```

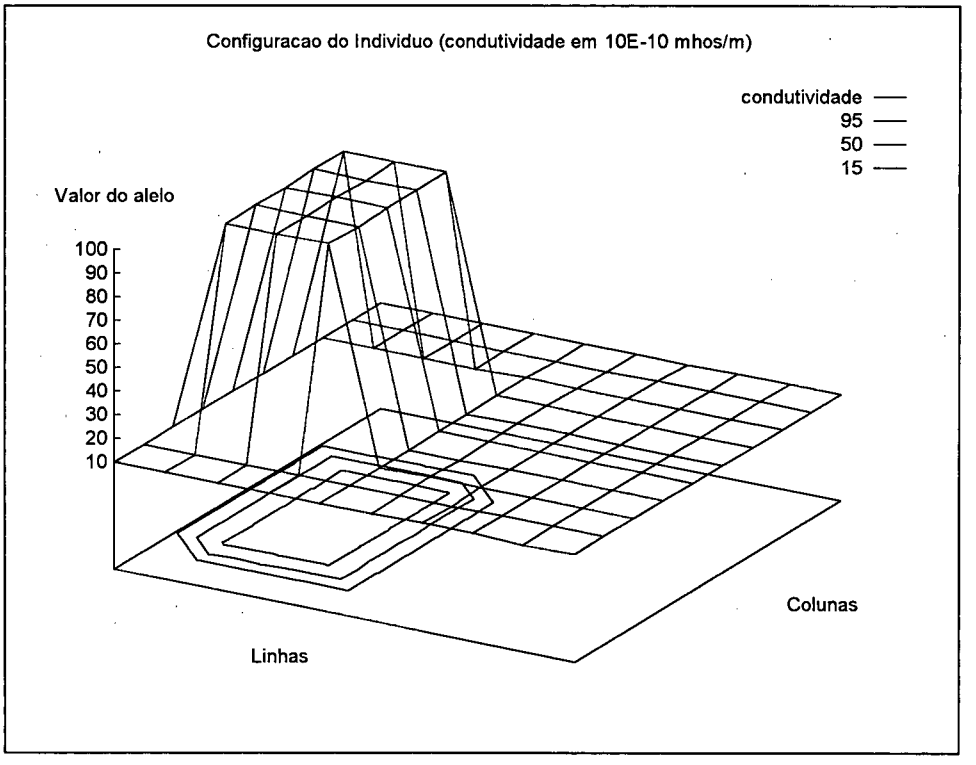


Fig. 8 : Representação da configuração 2 usada na inversão magnetotelúrica

3.4.1.1.3 Configuração 3

Esta é a configuração irregular por excelência. Novamente é um bloco de 22 prismas de condutividade 10 vezes maior do que a da rocha. Este bloco se inicia na superfície e se espalha nas diversas profundidades. Como nota, vale recordar que ele é contínuo. Sua representação numérica:

```
10.0 100.0 100.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0
10.0 100.0 100.0 100.0 100.0 100.0 100.0  10.0  10.0  10.0
10.0 100.0 100.0 100.0 100.0 100.0 100.0  10.0  10.0  10.0
10.0  10.0  10.0  10.0  10.0 100.0 100.0  10.0  10.0  10.0
10.0  10.0  10.0  10.0  10.0 100.0 100.0 100.0  10.0  10.0
10.0  10.0  10.0  10.0  10.0 100.0 100.0 100.0  10.0  10.0
10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0
```

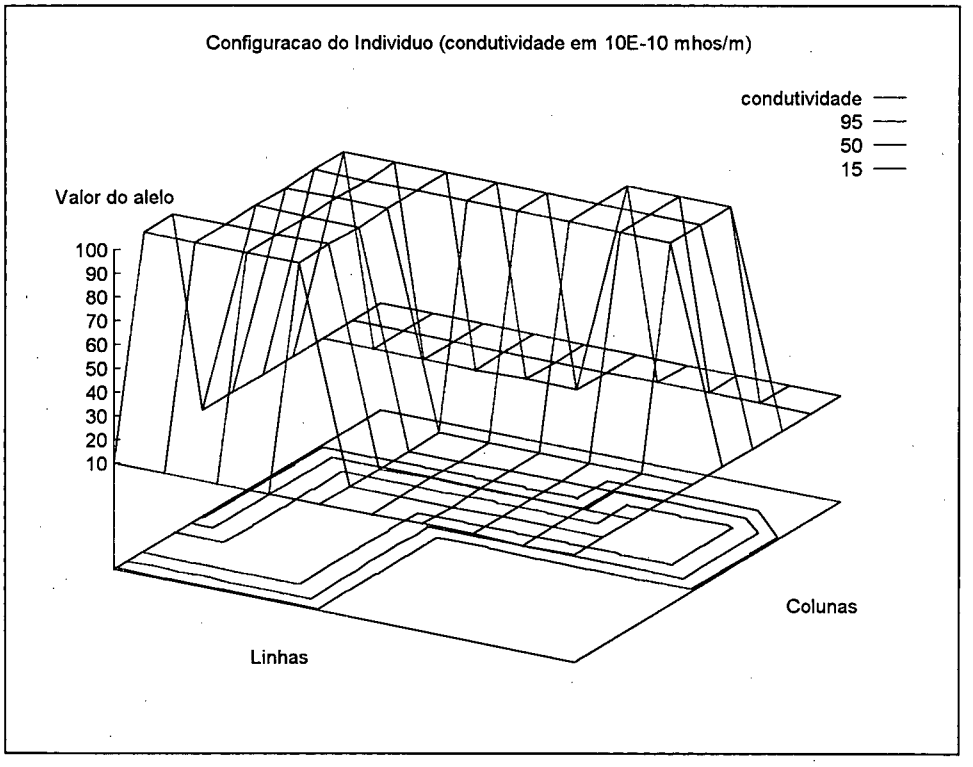


Fig. 9 : Representação da configuração 3 usada na inversão magnetotelúrica

3.4.1.1.4 Configuração 4

Esta é a configuração mais regular utilizada. É composta por rocha em toda a sua extensão, a menos de um único prisma que tem condutividade 10 vezes menor. Sua representação:

```
10.0  10.0  10.0  10.0   1.0  10.0  10.0  10.0  10.0  10.0
10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0
10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0
10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0
10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0
10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0
10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0
```

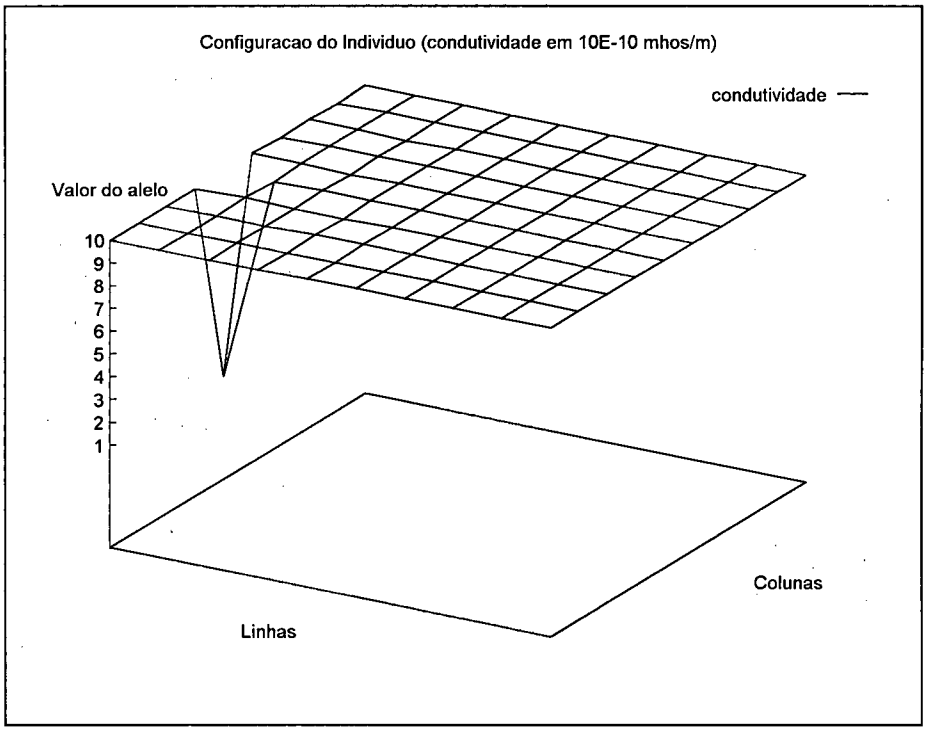


Fig. 10 : Representação da configuração 4 usada na inversão magnetotelúrica

3.4.1.1.5 Configuração 5

A configuração a seguir mostra um padrão inteiramente aleatório de condutividades. Foi colocada aqui apenas para uma investigação quanto ao comportamento dos métodos de solução frente a esta situação. Adianta-se que esta configuração não pôde ser resolvida por nenhum método.

52.0	76.0	68.0	76.0	68.0	52.0	39.0	53.0	4.0	94.0
46.0	52.0	68.0	52.0	6.0	84.0	22.0	76.0	84.0	5.0
39.0	84.0	53.0	68.0	22.0	53.0	84.0	84.0	52.0	46.0
39.0	6.0	22.0	68.0	84.0	68.0	22.0	22.0	68.0	54.0
68.0	6.0	6.0	76.0	6.0	94.0	94.0	5.0	53.0	68.0
22.0	46.0	53.0	46.0	94.0	68.0	22.0	6.0	94.0	68.0
53.0	76.0	84.0	84.0	4.0	46.0	76.0	52.0	6.0	39.0

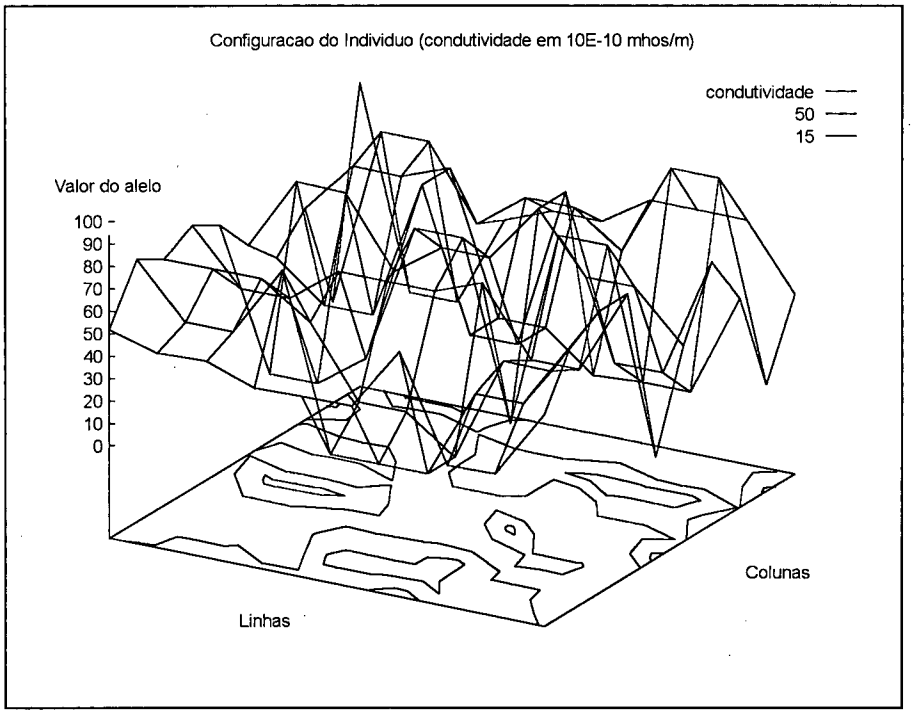


Fig. 11 : Representação da configuração 5 usada na inversão magnetotelúrica

3.4.2 Problema de Transmissão de Calor

Este problema começou a ser visto quando todo o tratamento para o Problema 1 anteriormente descrito já havia se dado. Desta maneira, considerando a similaridade existente em ambos os problemas inversos, buscou-se usar o mesmo ferramental, com as mínimas adaptações que se fizessem necessárias.

Neste problema, a matriz de materiais deve ter sempre um número ímpar de elementos e deve ser quadrada, por restrições de especificação da rotina de otimização utilizada e para permitir a construção do fractal de Sierpinski. A placa é composta de dois materiais distintos, cujos coeficientes de condutividade térmica são:

- material A tem condutividade térmica de 0,1 $W\ m^{-1}\ K^{-1}$
- material B tem condutividade térmica de 10,0 $W\ m^{-1}\ K^{-1}$

Uma placa -- não importa de qual dimensão -- constituída apenas por um dos materiais, terá condutividade igual à do material original. Entretanto, se composta por uma combinação entre os dois materiais, terá condutividade, variando entre 0,1 e 10 $W\ m^{-1}\ K^{-1}$ a depender da quantidade e posição de cada um dos materiais componentes.

A bancada de trabalho para este caso foi formada pelo modelo direto e por um engenho evolutivo. Aquele, um programa que, recebendo uma determinada configuração de placa, na forma de uma matriz bi-dimensional binária calculasse a condutividade térmica equivalente do conjunto. O engenho evolutivo foi muito parecido ao do problema geofísico, até porque, o objetivo foi estudar e validar as conclusões obtidas no estudo anterior.

Por similaridade com o problema de inversão magnetotelúrica, estabeleceram-se quatro configurações para o problema. Submetidas à ação do modelo direto estas geraram um resultado que passará a ser considerado como o padrão em cada configuração. Agora, definiu-se um engenho evolutivo que estabelece uma população de indivíduos candidatos à solução daquela configuração. Cada um dos candidatos é submetido ao modelo direto gerando este o valor de condutividade térmica para aquele candidato. Da comparação entre este valor e o padrão acima descrito, através de uma somatória de diferenças obtém-se a valoração (fitness) de cada um dos candidatos e este valor é quem conduz a busca evolutiva.

3.4.3 Problema da Difusão de Calor em Materiais Compostos

Este problema foi estudado em 3 instâncias distintas e de complexidade crescente. A primeira considera uma distribuição de condutâncias térmicas distintas ao longo de uma dimensão, a segunda trabalha com uma distribuição bi-dimensional e a terceira, tri-dimensional.

Em qualquer uma delas tem-se um material composto de fibra de carbono e de resina de epoxi, sendo que esta última faz o papel de impureza inserida no material original. Aplicando-se um pulso de calor em uma das extremidades do material observa-se o comportamento térmico ao longo do tempo na outra extremidade. O modelo direto, tem como informações a distribuição das impurezas no corpo do material e o pulso de calor aplicado, calculando e gerando o perfil de temperaturas da face oposta a que se aplicou o pulso de calor.

Para o caso 1D, tem-se uma parede de espessura 0.005m e composta por 10 "fatias" de um ou outro material. O modelo direto recebe 9 coeficientes de condutância térmica representando as 9 superfícies de contato entre as 10 fatias. Um pulso de calor é aplicado de um lado da parede e no lado oposto são tomadas 125 temperaturas, uma a cada 0,16 segundos. O processo todo toma 20 segundos (125 x 0.16 = 20).

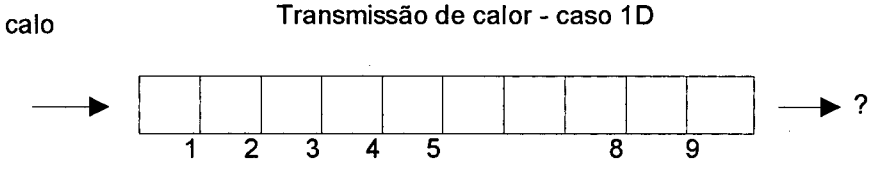


Fig. 12 : O problema de difusão de calor em 1 dimensão

O erro térmico para este caso é

$$E = \sum_{t=0}^{124} |Tc_t - Tp_t|$$

onde T_c e T_p são as temperaturas calculada e padrão, e a aptidão (fitness) é

$$F = \frac{1}{1.01^E}$$

Para o caso 2D, o modelo direto recebe 40 coeficientes térmicos, representando 10 superfícies de contato laterais e 4 superfícies longitudinais ao longo do eixo de transmissão do calor. O corpo tem 9 x 5 componentes. O modelo devolve -- para os 10 pontos -- um conjunto de 30 temperaturas tomadas ao longo de 15 segundos, com intervalo entre uma medida e outra de 0,5 segundos.

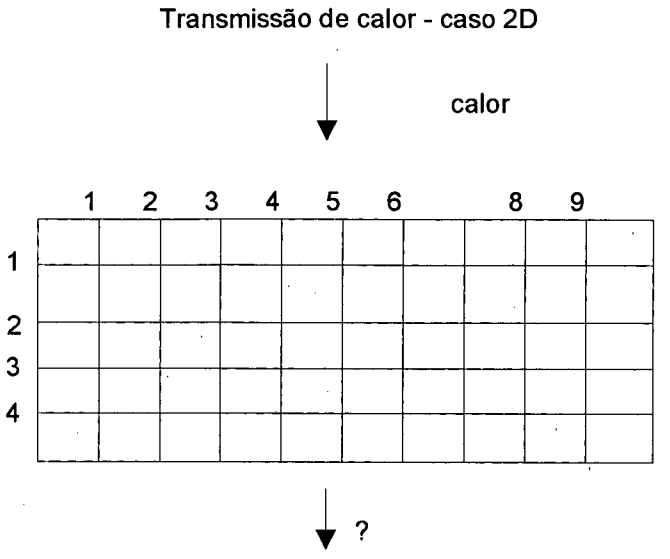


Fig. 13 : O problema de difusão de calor em 2 dimensões

O erro térmico para este caso é

$$E = \sum_{t=0}^{29} \sum_{m=1}^{10} \left| Tc_{mt} - Tp_{mt} \right|$$

e a aptidão (fitness) é

$$F = \frac{1}{1.01^{E}}$$

Para o caso 3D, o modelo direto recebe 400 valores de condutância, representando 4 planos de 10 x 10 condutâncias. O calor é transmitido ao longo dos planos. O modelo devolve para os 100 pontos da última superfície 10 temperaturas, tomadas ao longo de 5 segundos, com intervalos de 0,5 segundos.

Transmissão de calor - caso 3D

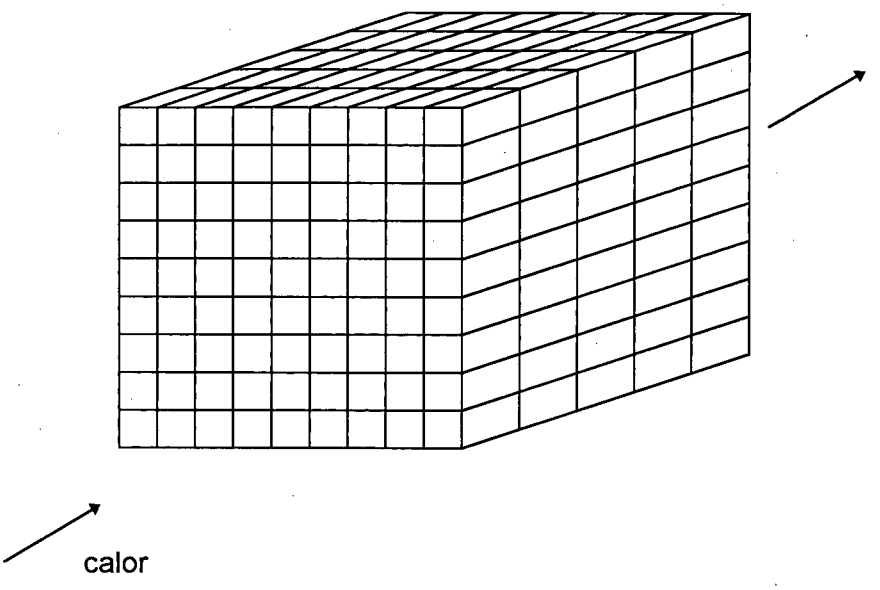


Fig. 14 : O problema de difusão de calor em 3 dimensões

O erro térmico para este caso é

$$E = \sum_{t=0}^{9} \sum_{m=1}^{10} \sum_{n=1}^{10} \left| Tc_{mnt} - Tp_{mnt} \right|$$

e a aptidão (fitness) é

$$F = \frac{1}{1.01^{E}}$$

Seguindo a estratégia adotada neste trabalho, estabelece-se uma determinada configuração para cada um dos casos. Submetida à ação do modelo direto este gera um resultado que passará a ser considerado como o padrão. Agora, define-se um engenho evolutivo que estabelece uma população de indivíduos candidatos aleatórios à solução do problema. Cada um dos candidatos é submetido ao modelo direto gerando este a configuração de temperaturas finais para aquele candidato. Da comparação entre estas temperaturas e o padrão acima descrito, através de uma somatória de diferenças obtém-se a valoração (fitness) de cada um dos candidatos e este valor é quem conduz a busca evolutiva.

4. Bancada de Trabalho

Descreve-se a seguir como foi implementada o modelo evolutivo a partir do pacote genérico utilizado. Uma discussão sobre as etapas percorridas e sobre as decisões tomadas nessas etapas acompanha a apresentação.

4.1 GALOPPS

The Genetic Algorithm Optimized for Portability and Parallelism System. Trata-se de um programa distribuído na rede Internet na forma fonte, possibilitando a seus usuários a adaptação ao problema que se quer resolver. O GALOPPS prevê a necessidade de ser incluída nele a função objetivo. Também podem ser incluídos os operadores evolutivos que se deseja. Abre-se a possibilidade do usuário codificar os seus próprios operadores. Neste caso, esta opção foi retardada até o último instante, já que sua adoção sempre implica perda de generalidade do modelo. Mas, diante de sua imprescindibilidade optou-se pelo que a literatura denomina *hibridização* (Davis, 1991). Trata-se do uso de características conhecidas do problema de maneira a acelerar ou simplesmente viabilizar o trabalho do engenho de computação evolutiva.

Utilizou-se a versão 2.37 em 96 e até meados de 1997, substituindo-se a partir de então esta versão pela 3.20. O programa é distribuído em modo fonte, escrito na linguagem ANSI C. Para sua utilização, deve ser convenientemente programada uma função objetivo (através de uma função C) que recebe uma solução candidata (indivíduo) a solução do problema e devolve um número real. Este valor é o *fitness* daquela solução candidata. Os demais tratamentos do processo evolutivo são providenciados pelas funções que compõem o pacote. São mais de 15.000 linhas de código C, fartamente autodocumentado. Uma característica determinante na escolha inicial foi a possibilidade do programa ser executado em um ambiente paralelo. Por razões de ordem prática, como veremos à frente, esta facilidade acabou não sendo utilizada, mas este foi um ponto relevante no início.

Este pacote tem sua ascendência no programa introduzido no livro de Goldberg (Goldberg, 1989) e que tinha o nome de SGA. Ele estava escrito em Pascal. Posteriormente, foi rescrito em C, dando origem ao programa conhecido como SGA-C. Existem inúmeras características interessantes no produto, que se encontram descritas em sua documentação (Goodman, 1996) e que não serão aqui tratadas. Serão mostradas a seguir apenas as funcionalidades de GALOPPS que tiveram especial interesse no tratamento dos problemas estudados nesta tese.

4.1.1 Alfabetos de diferentes cardinalidades

Em seus primórdios (SGA, SGA-C, ESGA, IPGA e GALOPPS) este produto trabalhava exclusivamente com alelos binários. Este procedimento encontra justificativa na afirmação de que códigos binários são suficientes para tratar qualquer problema[4]. Entretanto, há considerações de desempenho e de facilidade de uso. A partir da versão 3.20 este pacote aceita cardinalidades maiores que 2. Para o problema geofísico usou-se a maior cardinalidade admitida que é de 16383.

O uso de cardinalidades maiores que 2, garante que os operadores de mutação e de *crossover* não produzam resultados inválidos. Por exemplo, a mutação nunca geraria números fora do intervalo de existência dos alelos e a recombinação abster-se-ia de gerar cortes dentro dos bits formadores dos alelos, funcionando apenas em seus limites.

4.1.2 Uso de subpopulações

Um capítulo importante da computação evolutiva tem sido o de estudar o fenômeno de diversas populações sendo desenvolvidas em paralelo. De tempos em tempos estas populações podem trocar indivíduos como maneira de incrementar as bagagens genéticas de cada população. Isto tem sido apresentado com os nomes de subpopulações ilhadas ou estudo de nichos ecológicos. Uma excelente descrição deste tema se encontra na tese de doutorado de Mahfoud (1995).

GALOPPS aceita a utilização de subpopulações seja rodando em um único computador monoprogramado, seja em um multiprogramado ou em uma rede de computadores, quando de fato se teria o paralelismo real das populações. Exceto quanto a aspectos de desempenho, do ponto de vista dos resultados produzidos, não há diferença entre as 3 alternativas acima citadas.

No início deste trabalho, houve uma tentativa de se usarem subpopulações que trocariam material genético periodicamente. Foram testadas as opções de 5, 10 e até 20 populações rodando em paralelo. A cada certo número de gerações cada população cedia e recebia soluções (indivíduos) de seus vizinhos. Tipicamente a cada 10 gerações cada população cedia a 2 vizinhos 3 indivíduos: o de melhor *fitness* e outros dois aleato-

---

[4] Até porque, poder-se-ia argumentar que, seja qual for a cardinalidade, aparência, utilização, modo de operação, conjunto de operações admitidas etc etc, os alelos, quando gravados ou mesmo manuseados em memória nada mais são do que conjuntos de bits.

riamente escolhidos. Após o exame dos resultados concluiu-se que o eventual ganho conseguido com o paralelismo não foi suficiente para justificar o consumo de recursos. Dizendo de outro modo, havia um resultado mais adequado (preciso, robusto) quando se rodavam 20 execuções individuais, modificando-se apenas a semente aleatória do que quando se rodavam 20 subpopulações para a mesma instância do problema. Abandonou-se o uso de processamento paralelo com subpopulações pela inadequação do problema a este método.

4.1.3 Uso de arquivos de *checkpoint* e *restart*

Esta é uma facilidade puramente operacional (não acrescenta nenhum valor ao resultado numérico em si) mas que tem o seu valor. As maiores execuções do problema geofísico chegaram a ficar cerca de 3 dias rodando em um computador (em geral com uma carga de multiprogramação de 2 ou 3 instâncias do problema ao mesmo tempo). Tipicamente, geraram-se arquivos de recuperação a cada 10 gerações e em diversas ocasiões estes arquivos foram necessários para reiniciar um processamento abortado por quaisquer razões externas.

4.1.4 Capacidade de inicialização da população inicial

Esta facilidade também foi largamente usada. No início efetuou-se o teste de garantia de convergência do modelo. Este teste se inicia pela geração de (parte da) população inicial contendo valores próximos ou aderentes ao resultado que se busca (e que neste teste têm que ser de antemão conhecido). Posteriormente, e para comparações com o modelo de Ramos e Campos Velho (1996) com a finalidade de garantir igualdade de condições, a população inicial foi manipulada.

4.2 Solução Para o Problema de Reconstrução das Distribuições de Condutividade Geoelétrica

Este é um problema em estudo no âmbito do INPE (Instituto Nacional de Pesquisas Espaciais) que se iniciou por volta de 1996. Para efeitos desta tese, o estudo começou ao final de 1996. Enquanto no problema original era usada uma biblioteca de otimização baseada em subida da encosta (*hill climbing*) denominada NAG associado a técnicas de regularização, a proposta nesta tese foi estudar o mesmo problema, substituin-

do tal biblioteca pelo processamento usando computação evolutiva. Obviamente, o subproduto procurado com tal substituição é a generalização e a inferência de métodos e estratégias de uso da CE.

A primeira ferramenta usada foi o programa FORTRAN de obtenção do resultado direto do problema. Tal programa já existia e fora usado no estudo original. Ele recebe um conjunto de condutividades resultantes da distribuição regular de prismas de material homogêneo inseridos em um plano vertical à superfície terrestre. Cada prisma tem sua condutividade estabelecida e o conjunto de prismas tem as dimensões de 13 x 11 prismas. Destes 143, 88 estão no semi plano condutivo (isto é abaixo da superfície terrestre) e por condições de contorno, apenas 70 necessitam ser consideradas para o cálculo do campo magnético. O resultado do programa é a geração do campo magnético para 11 pontos (sempre com z = 0, isto é na superfície terrestre) e em cada um dos pontos, para 20 freqüências logaritmicamente espaçadas variando entre 0.0001 e 0.01 Hz. Em cada ponto e em cada freqüência obtém-se as componentes real e imaginária do campo. Desta maneira, o modelo direto de solução deste problema, recebe uma matriz de números reais de 7 linhas por 10 colunas representando condutividades, calcula o campo magnético da terra e devolve como resultado um conjunto de 440 valores assim discriminados:

- 20 freqüências
- 11 pontos
- cada valor é representado por parte real e imaginária

Dando como resposta final (20 × 11 × 2) o conjunto esperado de 440 valores. Antes de se iniciar o processamento evolutivo, determina-se um caso (que é o que se quer calcular), formado pelos valores de condutividade conhecidos de antemão, submete-se-os ao modelo direto, que gera os 440 valores de campo magnético e estes valores são guardados como padrão.

Para proceder à solução do problema inverso, geram-se indivíduos (7 x 10 valores) de maneira aleatória. Cada indivíduo é avaliado e gera os seus 440 valores específicos. Esses 440 valores gerados são comparados com o padrão que está armazenado desde o início do processamento. Da diferença entre os 440 valores calculados e este padrão, obtém-se, de alguma maneira, a aptidão do indivíduo.

4.2.1 Uma visão do trabalho

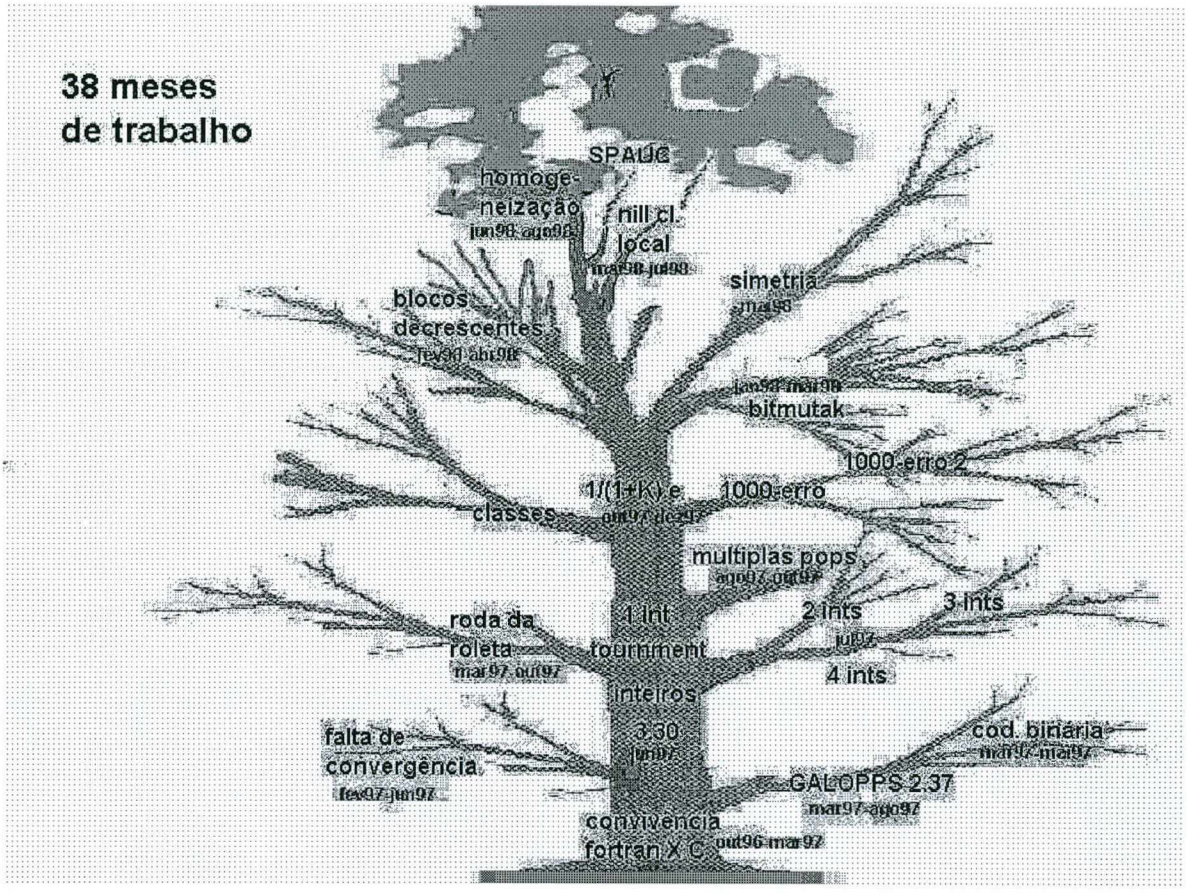


Fig. 15 : Método da tentativa e erro na solução do problema de inversão magnetotelúrica

A figura 15 mostra no tronco principal da árvore, as idéias e os desenvolvimentos que foram bem sucedidos, enquanto que nos galhos secos encontram-se as idéias que ao final de sua implementação mostraram-se insatisfatórias. Cada idéia está expressa por uma palavra, enquanto embaixo há uma data aproximada da época em que se investiu nessa abordagem. A diferença de volume da árvore (entre os galhos frutuosos e os secos) está aproximadamente correta e fornece noção da quantidade de trabalho que acabou não rendendo frutos, mas que de qualquer maneira foi produtiva ao valorizar as vertentes corretas do trabalho. A seguir, em grandes tópicos, a explicação de cada uma dessas abordagens.

4.2.2 Discussão sobre a função objetivo

A importância da função objetivo em qualquer processo envolvendo computação evolutiva está na obtenção de uma adequada aptidão apara o indivíduo. Fogel e Angeline (Fogel e Angeline, 1997) dizem "*Em qualquer computação evolutiva aplicada a um problema de otimização, o operador humano determina ao menos quatro aspectos da*

abordagem: representação, operadores, método de seleção e função objetivo. ...pode ser dito que o mais crucial destes quatro é a função objetivo, porque ela determina o resultado do operador (evolutivo) em termos quantitativos. A especificação de uma função objetivo imprópria pode levar à geração da resposta certa para a pergunta errada". Neste trabalho, a busca e seleção da função objetivo mais adequada passou por 3 etapas distintas.

4.2.2.1 Método das diferenças

Na primeira, adotou-se o modelo mais simplista possível:

$$f1 = 1000 - \sum diferenças$$

Ou seja, dado um indivíduo qualquer (70 condutividades), ele era processado pela função objetivo, gerando 440 medidas de campo elétrico. Essas medidas eram comparadas com os 440 valores correspondentes ao indivíduo que se buscava reconstituir. Dessa comparação resultavam 440 diferenças que eram somadas (em valor absoluto). A subtração de 1000 transformava um problema de minimização em maximização (exigência do GALOPPS). Este resultado final era o *fitness* daquele indivíduo original.

4.2.2.2 Método da categorização

$f2 =$ separação por classes atribuindo-se pontos a cada indivíduo

Na seqüência, tentou-se o desenvolvimento de nova função objetivo. A idéia era pontuar cada cromossomo, em função de quão perto ele estava do resultado esperado. Cada cromossomo dava origem a 440 valores e cada um desses era comparado com um dos 440 valores padrões já armazenados.

Se o indivíduo em análise se afastasse menos de 5% do padrão, (medido pela somatória das diferenças de campo elétrico -- tal como em *f1* acima), o cromossomo ganharia 5 pontos, se se afastasse 10%, ganharia 3 e se se afastasse até 20% ganharia 1 ponto. Se se afastasse mais de 20% não ganharia nada. Note-se que todos os valores aqui (5%, 10% e 20%, bem como 5, 3 e 1 pontos) eram parâmetros ajustáveis. Por exemplo, em nova rodada, eles foram mudados para 2%, 4% e 6%.

Nesta segunda abordagem, em uma rodada típica, processaram-se 100 gerações, e o melhor resultado foi 1941 pontos, o que informa que o melhor cromossomo assim se afastava do padrão. Percebe-se por este resultado típico que o desempenho da função objetivo *f2*, também deixou a desejar.

4.2.2.3 Método da pressão evolutiva variável

Para o cálculo desta função objetivo, utilizou-se um parâmetro variável denominado "pressão evolutiva" que permite, ao ser ajustado, determinar qual a velocidade esperada para a convergência[5]. O erro continuava sendo calculada como a diferença absoluta entre os valores de campo elétrico gerado e aquele previamente armazenado.

$$f3 = \frac{1}{(1+K)^{erro}}$$

K é um parâmetro com valor ligeiramente superior a 1. Particularmente, não houve variação de K, e seu valor definitivo acabou fixado em 1,01. A razão pela qual K acabou não sendo variado é que o operador de seleção utilizado foi o de torneio. Este operador, juntamente com os de classificação de dados (em contraposição aos ditos operadores proporcionais, tipicamente o de roda da roleta, por exemplo) são infensos à utilização de funções que apresentam probabilidade de seleção proporcional ao fitness. Para os métodos de classificação o que importa é apenas a determinação de quem é o fitness vencedor dado um conjunto de 2 a n indivíduos. Isto foi expresso por Brickle (Brickle, 1997) "*a seleção por torneio é invariante à translação e ao escalonamento. Isto significa que o escalonamento ou a translação do valor de fitness não modifica o comportamento do método de seleção. Assim, técnicas de escalonamento como as usadas na seleção proporcional não são necessárias, simplificando a aplicação do método de seleção*".

Isso não significa que o operador de torneio não possa ter a sua pressão seletiva modificada, mas apenas que a eventual modificação é feita de outra maneira, ou no caso, variando o tamanho do torneio. Assim para um torneio pequeno, tipicamente 2, a pressão é a menor possível (menos que isso, em um hipotético torneio de 1 indivíduo, não haveria pressão nenhuma e a seleção seria aleatória), até um torneio cujo tamanho seja igual ao da população, quando então a escolha do melhor seria determinística, inexistindo aleatoriedade.

---

[5] Mantendo a analogia com a natureza, este parâmetro de pressão poderia ser comparado à abundância / escassez de alimentos. Quando faltam os alimentos, a pressão seletiva aumenta, e apenas os mais aptos sobrevivem. Quando os alimentos abundam, a pressão diminui, permitindo que outros além dos mais aptos sobrevivam.

A razão da expressão $(1+K)^{erro}$ estar no denominador de uma fração visa transformar uma minimização (a busca do menor erro possível) em maximização, o que é uma exigência do pacote GALOPPS.

4.2.3 Discussão sobre a representação de cromossomos

A tentativa aqui foi elucidar qual o melhor compromisso entre tamanho do cromossomo e precisão do resultado. O estudo iniciou testando o uso de cromossomos binários. Buscava-se preservar o modelo binário para seguir a bibliografia, que o recomenda. Rapidamente verificou-se que o modelo não convergia e que havia necessidade de refinar mais a abordagem do problema. Descartado o uso de algarismos binários surgiu a questão de quantos bits usar para representar um número real. A constatação inicial foi de que o GALOPPS possui a facilidade de receber e devolver números inteiros (item 5.1.1), variando entre 2 e 16384. Considerando que este número é o máximo representável, e supondo-se a necessidade de representar números maiores do que este, o caminho seria o de usar vários inteiros deste tipo para representar um único real. supondo então este R como sendo o número real buscado e I_1, I_2, ... I_n a coleção de inteiros usados para representar R. Lembrar que R varia entre 1 e 100, já que é o multiplicador da condutividade, que por sua vez varia entre 10^{-11} mhos/m e 10^{-9} mhos/m.

Tabela 8 - Obtenção do alelo a partir de conjunto de inteiros

Quantidade de inteiros	Fórmula de obtenção do valor do alelo
Usando um único inteiro I	$R = 1 + \frac{I}{\frac{16384}{99}} = 1 + \frac{I}{165.49494949}$
Usando dois inteiros: I_1 e I_2	$R = 1 + \frac{(I_1 \times 16384) + I_2}{\frac{16384^2}{99}} = 1 + \frac{(I_1 \times 16384) + I_2}{2711469.25252525}$
Usando três inteiros I_1, I_2 e I_3	$R = 1 + \frac{(I_1 \times 16384^2) + (I_2 \times 16384) + I_3}{\frac{16384^3}{99}} = 1 + \frac{\ldots}{44424712233.3737}$
Usando quatro inteiros I_1, I_2, I_3 e I_4	$R = 1 + \frac{(I_1 \times 16384^3) + (I_2 \times 16384^2) + (I_3 \times 16384) + I_4}{\frac{16384^4}{99}}$ ou $R = 1 + \frac{\ldots}{727854485231595.43}$

Seguiu-se uma escala crescente, iniciando com um único inteiro. Não houve convergência. Passou-se a dois inteiros e novamente não houve convergência. O teste de 3 inteiros também foi infrutífero. Finalmente chegou-se a 4 inteiros e em não havendo convergência, concluiu-se que a falta de convergência não era devida a problemas de precisão de números. Isto posto, a busca de melhorias no algoritmo para forçar a convergência foi feita em outros locais (basicamente através de hibridização do algoritmo) e com isso provisoriamente retornou-se a 1 único inteiro que era, das alternativas, a que mais simplicidade oferecia ao processamento e operação do modelo. De todo modo, agiu-se assim em obediência ao *princípio do alfabeto mínimo* (Goldberg, 1989) que diz: *Deve-se selecionar o alfabeto mínimo que permita a expressão natural do problema.* Posteriormente, com o sucesso da hibridização e conseqüente convergência do modelo, não houve necessidade de retornar aos múltiplos inteiros para representar um único alelo.

Para finalizar esta discussão, o alelo final do modelo, foi um número real variando entre 1 e 100, e que utilizava um inteiro do GALOPPS com valor máximo de 16384. Cada variação de 1 no inteiro do GALOPPS correspondia a uma variação de 0.00604285 no alelo real. O número real "x" foi calculado com a fórmula

$$x = 1 + y \times 0.00604285$$

onde "y" é o inteiro correspondente. A título de exemplo, veja-se quais os valores de y para determinados valores de x

Tabela 9 Exemplos de conversão dos valores do alelo

y	x
0	1
1489	10
8109	50
16383	100

4.2.4 Discussão sobre as taxas

Desde o início variaram-se as taxas de utilização do engenho evolutivo. Rapidamente pode ser constatado, que desde que não houvesse grande distanciamento dos valores recomendados na bibliografia o algoritmo tinha convergências próximas umas das outras. Como conseqüência desta constatação, deixou-se de fazer uma busca completa sobre as melhores taxas possíveis para o problema, mantendo-se constantes até o final do trabalho os parâmetros básicos que governavam o engenho evolutivo. A

tabela 10 a seguir, apresenta as principais recomendações da bibliografia quanto a parâmetros e os valores escolhidos aqui.

Tabela 10 - Parâmetros recomendados na literatura

Operador	Valor	Autor	Valor empregado aqui
taxa de mutação	0.001 0.01 entre 0.01 e 0.05	Bäck, 1996	0.03
	0.001	Mitchell, 1997	
	0.0333	Goldberg, 1989	
taxa de *crossover*	0.6 0.95 entre 0.75 e 0.95	Bäck 1996 e Goldberg, 1989	0.85
	entre 0.7 e 0.8	Mitchell, 1997	
tamanho de população	entre 50 e 100 30 entre 20 e 30	Bäck 1996	100
	200	Fogel, cit. Bäck,1996	
	100	Davis, 1991	
tamanho do torneio	entre 6 e 10 (para programação genética)	Brickle, 1997	15
	≥ 10 (hard) e 3 -5 (soft)	Fogel, 1997	
	10	Fogel, citado em Bäck, 1996	
tipo de *crossover*			uniforme e SPAUC
Número de gerações			150

4.2.5 Mono ou multipopulação

Discute-se aqui o uso de múltiplas populações com e sem paralelismo. Chegou-se a ter até 20 subpopulações rodando em paralelo, ainda que em um único processador serial, já que não se usaram mais máquinas rodando em paralelo. Tentou-se com 10, 14 e 20 subpopulações, com esquemas padrão de troca de indivíduos. Usando processos absolutamente idênticos e comparando o erro magnético de uma rodada em monopopulação com outra usando 20 populações, obteve-se que o erro melhorou de 0.65 (melhor indivíduo em uma única população), para 0.53 (melhor indivíduo em 20 subpopulações). Registre-se que a melhora não foi significativa, já que o objetivo nesta etapa do trabalho era alcançar um erro magnético inferior a 0.1. Por esta razão, e considerando o consumo de recursos de rodadas multipopulações esta abordagem foi abandonada.

4.2.6 Função de mutação "sempre certa"

Com a finalidade de analisar a convergência do modelo, em determinada fase do trabalho foi desenvolvida uma nova função de mutação de bit, chamada BITMUTAK.C, que, dependendo de um parâmetro compilado junto, chamado probabilidade de acertar, ora chamava a função de mutação convencional do GALOPPS (bitmutat), ora chamava uma nova que colocava o valor esperado para o resultado que se pretendia alcançar. Por exemplo, se na posição x,y deveria aparecer um valor w, quando a função de mutação sempre certa fosse chamada nesta posição, ela geraria o valor w. Com esta nova função, pode-se forçar a convergência, variando-se o valor do parâmetro.

Quando a probabilidade de mutação sempre certa era 1 (100% dos casos), obteve-se uma convergência em 12 gerações. Quando ela foi usada com um valor 0.5, a convergência demorou cerca de 400 gerações para aparecer. Esta medida é uma particularização do assim chamado "*takeover time*" introduzida por Goldberg e Deb em 1991 (Brickle, 1997). O *takeover time* é o número de gerações gastas para que um indivíduo ótimo se espalhe preenchendo 100% das ocorrências na população.

Mediante o uso desta mutação foi possível conduzir testes de taxas e operadores, concluindo-se que elas influem muito pouco sobre a obtenção do resultado. Por exemplo, conduziu-se um teste em que 4 rodadas iguais foram disparadas. A única diferença era quanto ao operador de *crossover*. Após 300 gerações, as 4 apresentavam o mesmo valor.

4.2.7 Simetria

A primeira instância estudada e resolvida (o caso 2) apresentava uma certa simetria. Fez-se a hipótese de que essa simetria era característica do problema e portanto inicialmente o modelo usava a seu favor a simetria notada. Quando se passou a uma outra instância do problema, veio a decepção: a simetria que havia no problema anterior era específica daquela instância e não do problema, com o que, não poderia ser generalizada. O esquema de simetria diminuía o tamanho do cromossomo, replicando partes (supostamente simétricas) apenas imediatamente antes da passagem pela função objetivo. Nesta abordagem, o cromossomo teria 7 linhas por 6 colunas. Antes de submetê-lo à função objetivo (que esperava receber 10 por 7), fazia-se uma expansão do mesmo, seguindo o critério:

coluna -4: igual a coluna 5
coluna -3: igual a coluna 4
coluna -2: igual a coluna 3

coluna -1: igual a coluna 2

coluna 1: preservada a coluna 1 original

colunas 2, 3, 4, 5 e 6: idem idem.

O objetivo era o de diminuir o tamanho do cromossomo e conseqüentemente baixar a pouco mais da metade:

a) o espaço de busca

b) a quantidade de chamadas à função objetivo

O item (a) garantiria uma convergência em menor número de chamadas e ambos garantiriam um tempo menor de processamento para cada geração.

Como não houve sucesso, a abordagem foi abandonada.

4.2.8 Especialização crescente

Outra idéia foi a de criar indivíduos crescentemente especializados. Achou-se durante algum tempo, que a convergência poderia ser criada caso não se começasse diretamente com o cromossomo completo e sim com uma parte dele, que iria sendo expandida à medida em que o processo evolutivo se desse, e só próximo do final do mesmo o cromossomo teria seu tamanho completo.

Surge então a figura do indivíduo denominado "fatia". Isto é, ele tinha o mesmo valor nos 10 valores de uma dada linha do cromossomo. Quando usado em conjunto com a idéia de simetria, acima descrita seriam apenas 6 valores. Num primeiro estágio todas as condutividades teriam o mesmo valor. Uma justificativa, esta de ordem física, sugeriria que as condutividades existentes ocorreriam em blocos. Embora esta idéia não tenha tido sucesso, ela é antecessora da idéia de SPAUC, posteriormente desenvolvida.

Associado a este indivíduo, haveria um *crossover* uniforme "em linhas", fazendo com que, no cruzamento cada linha pudesse vir de um ascendente diferente. Isto tudo ocorreria por um número dado de gerações. Quando estas se encerrassem, a população seria guardada, para posterior recuperação pelo programa responsável pelo próximo ciclo.

Quando um novo ciclo fosse iniciado, antes de qualquer coisa, os indivíduos seriam reformatados, o que consistiria em gerar um nova população, em que cada indivíduo seria gerado por um indivíduo da população antiga no qual o único valor, que se repetiria 10 vezes (6 vezes se com simetria) seria substituído por 2 valores que se repetiriam 5 vezes, cada um (com simetria, 3).

Havia 4 ciclos, e o que se descreveu era a passagem do ciclo 1 para o 2. Na passagem do 2 para o 3, esses 2 valores seriam substituídos por 4 valores, e na transformação seguinte, os 4 valores serão substituídos por 10 valores.

Na etapa 1, o novo valor é mutado do valor velho segundo uma distribuição baseada em Poisson (ou seja, alta probabilidade de se obter um valor novo próximo do valor antigo e pequena probabilidade de se obter um valor novo distante do valor antigo).

Em resumo teve-se:

Tabela 11 Resumo do funcionamento da especialização crescente

	Com simetria		Sem simetria	
Na etapa	Quantidade de valores	Repetidos em	Quantidade de valores	Repetidos em
zero	1 valor	6 campos	1 valor	10 campos
um	2 valores	3 campos	2 valores	5 campos
dois	3 valores	2 campos	4 valores	2 e 3 campos
três	6 valores	não há repetição	10 valores	não há repetição

Este esquema era trabalhoso, pois em vez de um único conjunto de programas, havia 4 deles diferentes. Cada um destes possuía seu próprio operador de *crossover* e de mutação, já que havia que se tratar de número de variáveis distintas e crescentes. A despeito de todos este tratamento especializado, o melhor resultado que se conseguiu para o erro magnético foi de 0,4. Como o objetivo era um erro menor do que 0,1, este processo foi abandonado.

4.2.9 Porque é necessário hibridizar

Diante de tantos insucessos, uma vertente do trabalho foi tentar entender porque deixava de haver convergência. Fez-se um teste, que em visão retrospectiva pode ser considerado como o início da solução do problema. Em outras palavras, finalmente chegou-se ao âmago da questão "porque não há convergência ?"[6]. Eis a descrição do teste. Supondo que se busca o indivíduo cuja condutividade é (caso 2):

```
10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0
10.0  10.0 100.0 100.0 100.0 100.0 100.0  10.0  10.0  10.0
10.0  10.0 100.0 100.0 100.0 100.0 100.0  10.0  10.0  10.0
10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0
10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0
10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0
10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0  10.0
```

[6] Este fato se deu em 27 de fevereiro de 1998, uma sexta feira.

cujo erro magnético, por hipótese é 0.0. Comparado com o melhor indivíduo até aqui obtido, usando todas as técnicas acima descritas e obtido após 1100 gerações de uma população formada por 14 sub-populações, após 5 dias corridos de processamento era

7.4	9.3	12.8	7.6	13.4	6.6	14.5	9.5	10.4	10.8
19.9	15.1	99.5	88.6	98.2	84.3	99.9	16.2	6.4	9.2
1.3	11.4	93.0	99.0	99.4	99.8	23.3	29.9	5.6	46.4
16.6	45.4	3.2	29.2	6.9	37.6	87.5	5.3	15.5	1.3
12.8	14.0	24.4	11.6	10.2	20.1	17.4	11.1	12.4	19.4
10.6	11.7	78.5	4.2	25.2	10.3	13.4	11.7	11.8	31.6
13.4	13.7	1.5	30.5	4.3	4.6	35.2	13.4	7.3	14.3

cujo erro magnético era de 0,54. Para estudar a sensibilidade intrínsica ao problema em estudo imaginou-se uma solução completamente correta (e portanto idêntica ao indivíduo do caso 2 acima citado), a menos de um único valor, que seria dividido por 2. Gerando-se uma solução *ad hoc*, composta por 69 valores iguais ao objetivo final e apenas um deles (o primeiro valor, marcado em negrito) valendo apenas 50% do objetivo, como

5.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0
10.0	10.0	100.0	100.0	100.0	100.0	100.0	10.0	10.0	10.0
10.0	10.0	100.0	100.0	100.0	100.0	100.0	10.0	10.0	10.0
10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0
10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0
10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0
10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0

verificou-se ter este indivíduo um erro magnético de 0,86, portanto muito maior do que o obtido em processo evolutivo e que claramente era pior do que este em termos de uma inspeção visual. Para conferir, basta comparar os 2 individuos acima.

Meditando sobre este resultado, percebeu-se que a falta de convergência não era um defeito do processo evolutivo, e que a despeito do que quer que se fizesse neste, se nenhum conhecimento adicional do modelo fosse introduzido, nunca se obteria o resultado buscado (que repetindo, era de encontrar um erro magnético inferior a 0,1 para problemas ainda sem ruído gaussiano). Estava-se diante da instabilidade apontada por C de Mol (1992) como característica dos problemas inversos. Em outras palavras, a instabilidade inerente do modelo forçou o uso de conhecimento do problema para obter uma solução que fosse adequada. Tal instabilidade é caracterizada pela variação brusca do erro quando um único valor na entrada é ligeiramente modificado. Poder-se-ia afirmar ter este problema características caóticas (Lorenz, 1993).

Para estudar a instabilidade do modelo, refez-se este procedimento 70 vezes, uma para cada valor da matriz de condutividades. Em cada uma das 70 vezes, deixa-

ram-se 69 valores absolutamente corretos, e apenas um deles (em cada vez) foi dividido por 2. Calculou-se o erro em cada um dos casos hipotéticos. Na tabela 12 tem-se o resultado deste teste. Em cada célula da tabela 12 encontra-se o erro magnético de um indivíduo que tivesse todas os valores corretos, a menos do valor correspondente à célula em análise. Esta condutividade seria apenas a metade do valor correto.

Tabela 12: Erros caso uma única célula seja dividida por 2

0.86	1.20	0.82	0.46	0.46	0.46	0.82	1.20	0.87	0.45
0.42	0.53	0.69	0.36	0.36	0.36	0.69	0.53	0.42	0.22
0.21	0.24	0.12	0.15	0.15	0.15	0.12	0.24	0.21	0.11
0.10	0.10	0.11	0.02	0.03	0.03	0.11	0.10	0.10	0.06
0.29	0.20	0.07	0.04	0.04	0.04	0.07	0.20	0.29	0.15
0.06	0.05	0.03	0.01	0.01	0.01	0.03	0.05	0.06	0.03
0.03	0.03	0.02	0.01	0.01	0.01	0.02	0.03	0.03	0.02

Segue-se um exemplo da leitura da tabela acima: o número 0,69 encontrado na linha 2, coluna 3 é o erro magnético encontrado se o elemento da linha 2, coluna 3 do objetivo buscado (e apenas ele) fosse dividido por 2, mantendo-se os outros 69 corretos.

Em linguajar da computação evolutiva, diz-se que o problema apresenta epístase, isto é, há uma clara interdependência entre os diversos *lócus* do cromossomo. A epístase pode ser dividida em pleiotropia (um único gene afeta diversas características fenotípicas) e poligenia (uma única característica fenotípica é afetada pela iteração simultânea de diversos genes). Há boas descrições da epistasia em (Beasley, Bull e Martin,1993a) e em (Harvey, 1992).

4.2.10 Obtenção da solução do problema

Baseado no fato acima, abandonou-se a busca de melhorias exclusivamente sobre o engenho evolutivo, passando a considerar também – e principalmente – melhorias advindas do conhecimento do processo geofísico. Esta iniciativa finalmente conduziu à solução do problema que se encontra descrita em detalhe no capítulo 5.

5. Resultados para o Problema da Reconstrução das Distribuições de Condutividade Geoelétrica

5.1 Introdução

Descreve-se a solução de um problema inverso usando computação evolutiva que só teve solução completa quando se lançou mão do projeto e implementação de operadores evolutivos *ad-hoc*. Esta não é abordagem inédita. De acordo com Davis, isso pode ser feito usando 3 princípios: 1. Algoritmos adaptados ao problema podem ser usados para gerar os indivíduos de parte da população inicial. Usando-se elitismo, garante-se que o engenho evolutivo não terá desempenho pior do que o algoritmo já existente; 2. Incorporando heurísticas ou procedimentos do algoritmo já conhecido aos operadores genéticos; 3. Enriquecendo o algoritmo evolutivo com esquemas especializados de codificação.

Kodyalama e DeStefano (1996) dizem que *a etapa mais criativa no processo evolutivo é a criação de operadores genéticos apropriados para o problema particular em estudo*, e seguindo este conceito, eles desenvolveram um operador específico do problema (chamado *switching*) e obtiveram bons resultados num contexto de projeto de componentes de satélites.

Similarmente, Bruns (1993) chama os operadores evolutivos que utilizam conhecimento do problema para melhorar seu desempenho de *knowledge-augmented genetic operators.*

5.2 Resultados

Segue-se a descrição dos resultados obtidos nas diversas execuções do modelo tradicional, denominado "otimização via gradiente mais regularização" e evolutivo. Ênfase especial é dada aos parâmetros utilizados em cada execução ou seqüência de execuções. Em cada uma é necessário determinar previamente a seguinte coleção de parâmetros:

- **Para o engenho evolutivo**

a) Configuração utilizada, ou seja qual o indivíduo que se busca reconstituir (obtido das 5 configurações acima descritas).

b) Distribuição dos valores formadores dos indivíduos da população inicial, ou seja quão aleatória era a distribuição inicial dos indivíduos da população.
c) Aplicação de ruído. Sua existência, a quantidade de medidas, o valor do ruído gaussiano aplicado e o tipo de resultado do processamento de ruído utilizado.
d) Periodicidade de processamentos de otimização externa ao engenho evolutivo.
e) Número de ciclos de otimização. Componentes de cada ciclo (busca local e homogeneização). Quantidade de tentativas de homogeneização.
f) Elitismo. Presença garantida do melhor indivíduo na próxima geração.
g) Operador de crossover utilizado. Pode ser SPAUC ou crossover uniforme. Qual a probabilidade de cada um. No caso de SPAUC, quantidade de linhas de corte e quantidade de colunas de corte. Probabilidade dos blocos resultantes de SPAUC sofrerem homogeneização para o descendente 1 e para o descendente 2.
h) Parâmetros de rodada. Número máximo de gerações, tamanho da população, probabilidade de crossover e de mutação, tamanho do torneio e semente aleatória.
i) Resultados obtidos. Número real de gerações até a estabilização do resultado. Número de chamadas à função objetivo. Desempenho (em forma gráfica) do engenho evolutivo. Melhor indivíduo em forma gráfica. Erro magnético do melhor indivíduo. Erro condutivo do melhor indivíduo.

- **Para a otimização via gradiente mais regularização**

a) Configuração utilizada
b) Distribuição dos valores formadores dos indivíduos da população inicial
c) Aplicação de ruído. Sua existência e o valor do ruído gaussiano aplicado.
d) Parâmetros aplicados na regularização. Quais os valores de γ_1 e γ_2.
d) Resultados obtidos. Número de chamadas à função objetivo. Erro magnético do melhor indivíduo. Erro condutivo do melhor indivíduo.

Usar-se-á a seguinte tabela acompanhando cada execução

Tabela 13: Parâmetros a serem pesquisados em cada rodada	
Parâmetro	**Explicação**
Configuração utilizada	Qual das 5 configurações acima mostradas se buscou reconstituir
Distribuição da população inicial	Aleatória ou igual ao valor de contorno (rocha): 10 x 10^{-11} mhos/m
Ruído	Aplicação de ruído gaussiano de desvio padrão igual à percentagem aqui expressa sobre a medida original com média igual a zero.
Otimização externa	Quantidade de gerações entre chamadas de otimização. E, quando a otimização é invocada, quantos ciclos de busca local e quantos de homogeneização.
Elitismo	Sim ou não
Crossover	Tipo e taxas associadas
Parâmetros da rodada	Tamanho da população, número de gerações, taxas de recombinação e de mutação
Resultados obtidos	Erro magnético do melhor indivíduo. Erro condutivo do melhor indivíduo; quantidade de chamadas à função objetivo (unidade é 1000 chamadas)

5.2.1 Sobre Operadores Específicos ao Problema

Diversos autores têm abordado o problema de operadores evolutivos adaptados ao problema em estudo. O que se perde em generalidade deixando de se poder reutilizar código é compensado pelo aumento do desempenho do engenho evolutivo.

Guerreiro e colaboradores (1998) afirmam *"em GA está-se livre para criar outros"*, e coerentes com esta afirmação, eles apresentam a idéia de usar 3 operadores de recombinação diferentes, com taxas distintas para cada um deles.

Kajiwara e Nagamatsu (1996), em um contexto de otimização estrutural sugere que o engenho evolutivo faça a conversão de um cromossomo unidimensional para outro tridimensional de acordo com o problema e que as relações de contiguidade e vizinhança no espaço sejam preservadas no cromossomo. Criaram para isso dois operadores de crossover, denominados *scooping out* e *cutting off*. Ambos com a idéia comum de gerar blocos tridimensionais que são intercambiados entre os ancestrais.

Cartwight e Harris (1993) introduziram um operador de crossover, a quem denominaram de UNBLOX, *(uniform block crossover)*. Ele se deriva de um operador de crossover de 2 pontos, devidamente adaptado a um cromossomo bidimensional (ainda que representado em apenas 1 dimensão). Na recombinação uniforme convencional, os pontos centrais de um cromossomo bidimensional que foi "esticado" têm probabilidade maior de fazerem parte do corte em comparação com os pontos que estão na periferia

do cromossomo. Este efeito tem sido chamado de *positional bias*. UNBLOX corrige esta anomalia, garantindo que todos os pontos do cromossomo tenham a mesma probabilidade de gerarem cortes. O operador SPAUC a seguir proposto, não sofre de *positional bias*, por não estar baseado na recombinação de 2 pontos e sim no *uniform crossover* (recombinação uniforme). Ele não trata o cromossomo bidimensional simplesmente esticando-o e sim, como uma entidade bidimensional completa.

Tanaka e colaboradores (1993), mostram um esquema de codificação de dados de geomagnetismo que agrega o conceito de distribuição espacial das correntes elétricas. Sugere-se a idéia de crossover em retângulos embora não haja maior desenvolvimento.

Matthew (1999), em apresentação tutorial disponível na Internet, ao descrever os operadores genéticos, comenta sobre um "operador de crossover para *arrays*". Exemplifica o conceito através de um caso tri-dimensional, que tem um único ponto em seu interior aleatoriamente escolhido. Este único ponto determina 8 regiões contíguas que são trocadas pelos ancestrais. Veja-se no desenho, extraído de Matthew (1999), o resultado de um "operador de *crossover* para *arrays*" envolvendo um ancestral claro e outro escuro,

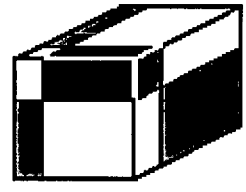


Fig. 16: Operador de *crossover* para *arrays* de Matthew (1999)

Finalmente, Mitchell (1997) e Bäck (1996) descrevem diversas implementações de crossover. São citados o *segment crossover* (no qual o número de pontos de corte não é fixo, mas pode variar em torno de um valor esperado, talvez seguindo uma distribuição de Poisson); o *shuffle crossover* (no qual os ancestrais são embaralhados de alguma forma, é feito a recombinação e o resultado é desembaralhado); o *punctuated crossover* (o indivíduo carrega informação de número e local de cortes. Por fazerem parte do genótipo tais informações também são objeto de otimização).

De qualquer maneira, a geração de novos operadores genéticos não é um problema encerrado. Como citado por Mitchell (1997) todos os estudos se referem a conjuntos pequenos de dados de teste e pior -- diferentes estudos conduzem a resultados conflitantes. Como diz Bäck (1993) "*durante cada uma das principais conferências em algoritmos genéticos é esperado um conjunto de novos operadores, em especial quando uma representação não tradicional (para o problema) é usada* ".

Os resultados numéricos aqui descritos foram obtidos usando-se três operadores genéticos especialmente construídos: SPAUC, que vem a ser uma generalização bidimensional para o operador de crossover uniforme, a busca local que privilegia descontinuidades e a homogeneização que por seu turno busca " espalhar" valores que quando distribuídos nas cercanias de um ponto aumentam o valor da função objetivo. Estes dois últimos, em certo sentido, atuam de maneira antagônica. Os resultados mostraram, que a despeito disso, o uso concomitante dos dois operadores conduziu a bons resultados.

5.2.1.1 Spatial Uniform Crossover

Segundo Boschetti e colaboradores (1996), "*um dos principais problemas na aplicação de algoritmos genéticos a problemas geofísicos é a sua alta dimensionalidade*". É o caso na presente aplicação, mesmo considerando que ela está baseada em dados sintéticos. De fato, como um indivíduo solução é composto por 70 números reais, cada um deles sendo representado em 14 bits, o que levaria a um espaço de soluções da ordem de 2^{980} o que convertendo para a base decimal, apresenta como resultado algo próximo de 1.02×10^{295}.

Um engenho evolutivo "canônico" num problema sintético, sem ruído, não apresentou resultado adequado. Começou-se então a busca por operadores *ad-hoc*. Em experiências prévias verificou-se que para este problema o operador de crossover uniforme tem desempenho superior ao da recombinação de 2 ou n pontos. Assim, quando surgiu a necessidade de propor um novo e melhor operador de crossover, este foi uma generalização da recombinação uniforme, apesar de que seguem funcionando juntos. A escolha entre SPAUC e recombinação uniforme se dá pelo estabelecimento de uma probabilidade p escolhida pelo usuário que fica fixa ao longo de uma dada execução. Assim, quando p=1.0 todas as chamadas ao operador são atendidas por SPAUC. Quando p=0.0 são atendidas pela recombinação uniforme.

O operador SPAUC considera cada indivíduo como uma entidade bidimensional. Desta maneira, ao intercambiar material genético regiões retangulares inteiras da solução candidata são trocadas. O operador implementa o conceito de vizinhança tanto na vertical quanto na horizontal. Com isso evitam-se dois problemas que ocorrem quando um indivíduo bidimensional é transformado em unidimensional: a) dois vizinhos na vertical deixam de sê-lo quando o cromossomo é "esticado" e b) elementos originalmente afastados (por exemplo, o último valor de uma linha e o primeiro da linha seguinte) ficam juntos quando o cromossomo é "esticado".

Para implementar este operador, tem-se o seguinte algoritmo:

a) São gerados n cortes horizontais em posições aleatórias no indivíduo;
b) São gerados k cortes verticais em posições aleatórias no indivíduo (O número de regiões no cromossomo é de r = (n+1) × (k+1));
c) É gerada uma máscara V, com tamanho r, de valores binários aleatórios;
d) Dados dois ascendentes que vão sofrer a recombinação, são gerados os descendentes sujeitos à regra: regiões que têm máscara igual a zero herdam seus valores do ascendente 1. Regiões com máscara igual a 1, herdam seus valores do ascendente 2. Note-se que quanto maior o comprimento de V, mais SPAUC se aproxima de um operador de recombinação uniforme convencional.

A seguir, exemplifica-se uma possível aplicação de SPAUC. Inicia-se gerando valores aleatórios que indicarão as linhas de corte no cromossomo. A quantidade de cortes é configurável. Supondo que se tenha gerado dois cortes, em 3 e 4 eis como ficaria um indivíduo A1, candidato a sofrer SPAUC

		a	a	a	a	a	a	a	a	a	a
		a	a	a	a	a	a	a	a	a	a
		a	a	a	a	a	a	a	a	a	a
A1	=	a	a	a	a	a	a	a	a	a	a
		a	a	a	a	a	a	a	a	a	a
		a	a	a	a	a	a	a	a	a	a
		a	a	a	a	a	a	a	a	a	a

Segue o algoritmo gerando colunas aleatórias de corte. Supondo que sejam gerados três colunas em 3, 4 e 7. O mesmo indivíduo A1 ficaria

a	a	a	a	a	a	a	a	a	a
a	a	a	a	a	a	a	a	a	a

		a	a	a	a	a	a	a	a	a	a
A1	=	a	a	a	a	a	a	a	a	a	a
		a	a	a	a	a	a	a	a	a	a
		a	a	a	a	a	a	a	a	a	a
		a	a	a	a	a	a	a	a	a	a

Perceba-se que os cortes levaram a 12 regiões distintas dentro do indivíduo. Conseqüentemente, o vetor de máscaras típico da recombinação uniforme necessitará ter 12 bits, para indicar qual ascendente gerará cada uma das regiões. Supondo uma máscara igual a V = 1 1 0 1 1 0 1 0 1 1 1 0 , e supondo um segundo ascendente denominado A2 composto exclusivamente por valores "b", isto é

		b	b	b	b	b	b	b	b	b	b
		b	b	b	b	b	b	b	b	b	b
		b	b	b	b	b	b	b	b	b	b
A2	=	b	b	b	b	b	b	b	b	b	b
		b	b	b	b	b	b	b	b	b	b
		b	b	b	b	b	b	b	b	b	b
		b	b	b	b	b	b	b	b	b	b

A aplicação de SPAUC geraria os seguintes descendentes

		b	b	b	b	a	a	a	b	b	b
		b	b	b	b	a	a	a	b	b	b
		b	b	b	b	a	a	a	b	b	b
D1	=	b	b	b	a	b	b	b	a	a	a
		b	b	b	b	b	b	b	a	a	a
		b	b	b	b	b	b	b	a	a	a
		b	b	b	b	b	b	b	a	a	a

		a	a	a	a	b	b	b	a	a	a
		a	a	a	a	b	b	b	a	a	a
		a	a	a	a	b	b	b	a	a	a
D2	=	a	a	a	b	a	a	a	b	b	b
		a	a	a	a	a	a	a	b	b	b
		a	a	a	a	a	a	a	b	b	b
		a	a	a	a	a	a	a	b	b	b

5.2.1.2 Implementação de SPAUC

Considerando que se utilizou o programa GALOPPS como base para a computação evolutiva, houve a preocupação de programar SPAUC de maneira integrada a este programa. Como se disse acima, ele foi programado dentro da rotina de recombinação

uniforme, mantendo-se o mesmo protocolo de chamada e de devolução dos resultados para a função computacional.

No início da rotina, há uma variável denominada PROBSPAUC, cujo intervalo de variação é entre 0.0 e 1.0. Ela representa a probabilidade de chamada de SPAUC *vis-à-vis* recombinação uniforme. Quando ela for 1.0, 100% das chamadas ao operador de *crossover* serão atendidas por SPAUC e 0% por recombinação uniforme.

Seguem-se as variáveis LPP e CPP que determinam a quantidade de cortes horizontais e verticais que devem ser feitos no cromossomo. O local dos cortes é determinado de maneira randômica pelo algoritmo, mas a quantidade de cortes é estabelecida a priori.

Neste caso, $1 \leq LPP \leq 7$ e $1 \leq CPP \leq 10$.

Os locais de corte são estabelecidos pelas fórmulas

$$L_0 = 1 + random(7 - LPP) \text{ e } L_n = L_{n-1} + 1 + random(7 + n - (LPP + L_{n-1}))$$

5.2.1.3 Comparação entre SPAUC e Recombinação uniforme: Dados sem ruído

Foram feitas 3 medidas para a configuração 2, com inicialização da população de maneira aleatória, sem ruído, com 1 ciclo de busca local a cada 10 gerações, ciclo este composto de 5 etapas (mais um valor que variava em função do número da geração acrescentando-se uma unidade a cada 10 gerações) de busca local e 3 etapas de homogeneização. Presente o elitismo e utilizado a recombinação sob teste. Todos os parâmetros, com exceção do operador de recombinação (e da semente aleatória) foram mantidos iguais. Um resumo destes parâmetros está na tabela 14.

Tabela 14 - Descrição dos testes de comparação entre SPAUC e Recombinação uniforme

Parâmetro	Valor
Configuração utilizada	2
Distribuição da população inicial	aleatório
Ruído	0.0
Otimização externa	1 a cada 10 gerações (=0); local=5+gen/10;homo=3
Elitismo	1
Recombinação	1.0 SPAUC; L=2; C=3; F1=0.05; F2=0.05
Parâmetros da rodada	G=150; P=100; PC=0.85; PM=0.03; T=15

Os resultados alcançados, para 4 rodadas variando apenas a semente aleatória, foram

Tabela 15 : Resultados da comparação entre SPAUC e Recombinação uniforme (sem ruído)		
Rodada	Erro magnético	Erro condutivo
SPAUC 1	0.67	818.6
SPAUC 2	0.54	583.0
SPAUC 3	0.66	663.1
SPAUC 4	0.03	124.9
MÉDIA	0.47	547.4
Uniforme 1	0.59	764.7
Uniforme 2	0.83	987.4
Uniforme 3	0.28	733.3
Uniforme 4	0.56	860.4
MÉDIA	0.56	836.4

O melhor resultado por SPAUC foi obtido na rodada SPAUC 4 e é

```
10.0  10.0   9.8  10.2   9.8  10.2   9.7  10.1   9.9  10.1
10.0   9.9  98.4 100.0 100.0 100.0  94.0  10.6   9.5  10.4
10.0  10.0  99.9 100.0  84.5 100.0 100.0  10.4   9.4  10.9
10.1   9.9  10.2  20.5  58.9  21.8  11.1  10.1  10.2   9.8
10.0  10.0  10.0   9.7   7.4   8.3   9.9   9.6  10.0   9.7
10.2  10.0  10.0  12.9  26.4  11.6  10.3  10.2  10.0  10.8
 9.7  10.0  10.0   9.9   8.5   9.5  10.1   9.5   9.5   9.8
```

que pode ser assim visualizado

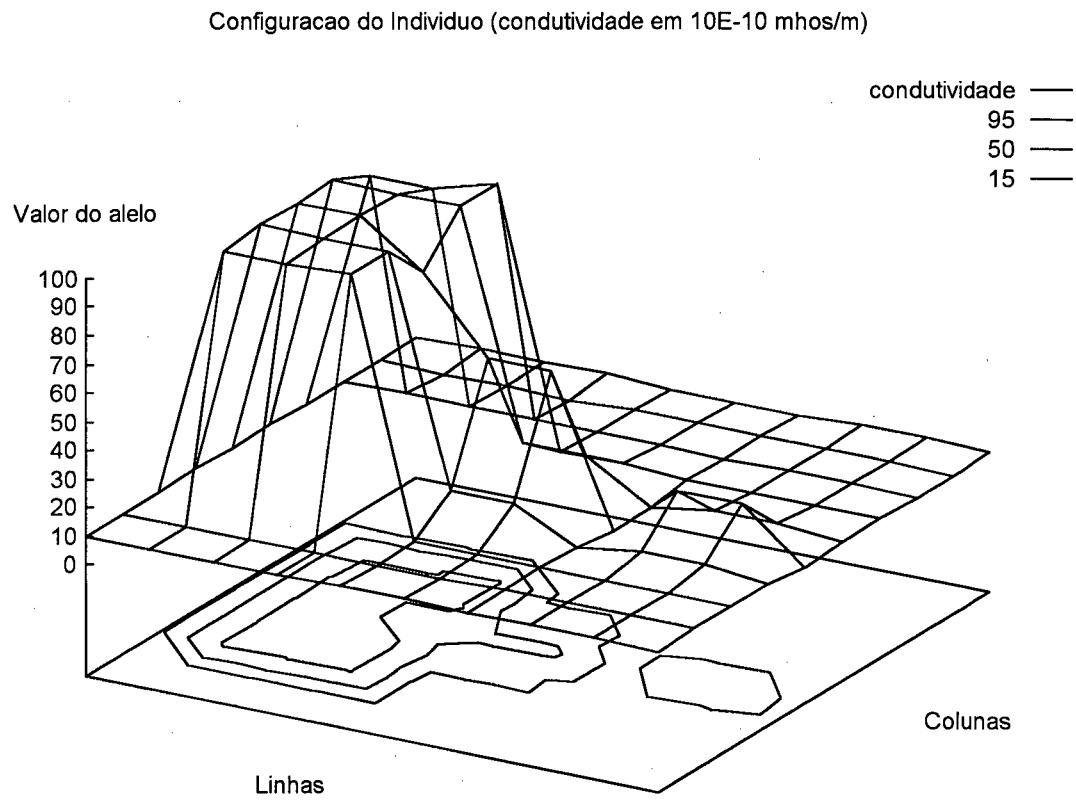


Fig. 17: Melhor resultado usando SPAUC, dados sem ruído

O melhor resultado para Recombinação uniforme foi obtido na rodada Uniforme 3 e foi

10.0	10.0	9.9	9.3	11.0	8.0	10.4	10.2	8.4	11.9
10.0	9.9	97.4	100.0	99.0	100.0	41.4	44.1	4.5	41.9
10.2	10.2	58.7	100.0	79.4	74.7	97.7	5.4	2.8	9.5
9.8	10.0	38.1	47.6	50.2	70.7	52.9	33.7	23.9	77.1
10.1	9.8	5.0	7.7	1.2	2.3	10.3	9.6	8.3	3.0
9.9	9.2	12.9	6.1	19.2	5.2	14.9	11.0	3.2	50.7
10.6	7.1	18.8	2.6	16.1	8.1	21.8	3.3	15.2	25.0

e pode ser assim visualizado

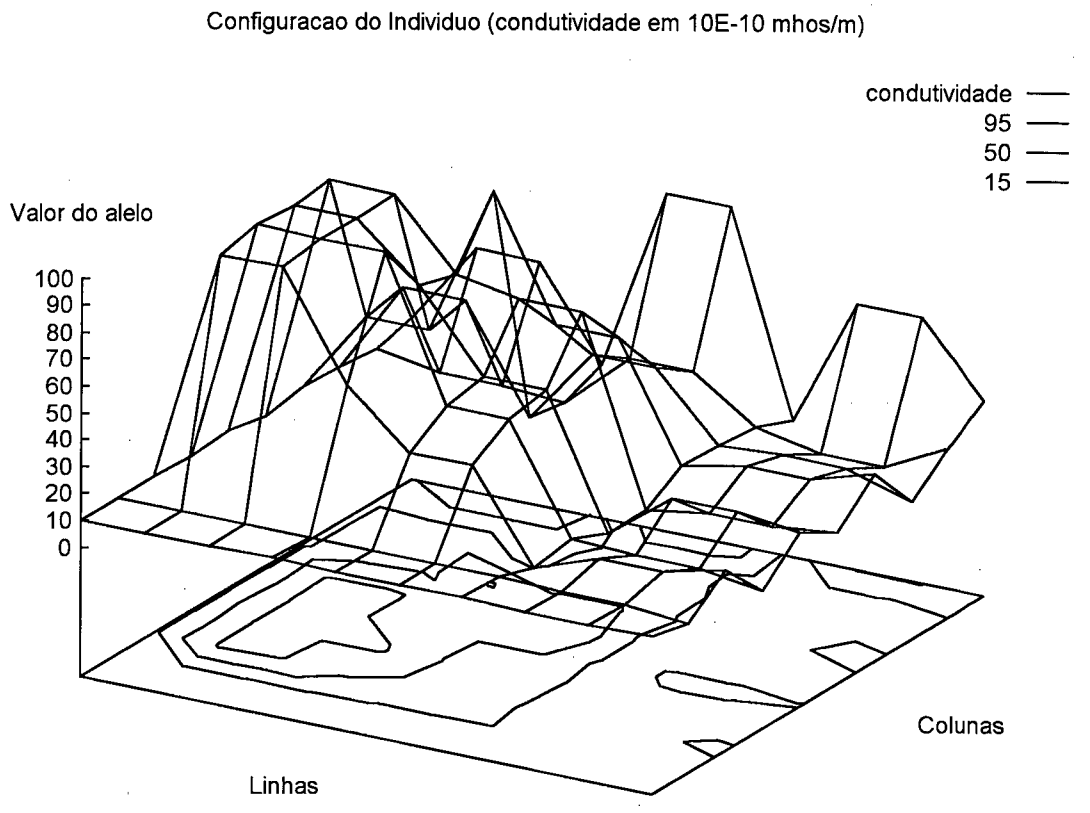


Fig. 18: Melhor resultado usando recombinação uniforme, dados sem ruído

Pode-se observar o desempenho melhor de SPAUC *vis-à-vis* recombinação uniforme. Esta melhoria é tanto qualitativa (124.9 versus 733.3 de Erro condutivo) quanto visual, bastando olhar os indivíduos acima. Finalmente, nas médias do teste, tem-se para os erros condutivos 547.40 para SPAUC e 836.45 para Recombinação uniforme.

5.2.1.4 Comparação entre SPAUC e Recombinação uniforme: Dados com ruído

Seguiu-se agora um teste idêntico ao anterior, só que com adição de ruído gaussiano. A configuração buscada é a mesma do caso acima, e os parâmetros são igualmente idênticos com exceção do ruído que foi adicionado.

Este é distribuído normalmente (distribuição gaussiana) com média igual a 0 e desvio padrão igual a 1% da medida original. São aplicadas três adições de ruído à me-

dida original, obtendo-se três medidas ruidosas. Finalmente, o valor a ser usado no processamento é a média entre as três. A tabela 16 resume estes parâmetros

Tabela 16 : Resumo dos parâmetros na comparação entre SPAUC e Recombinação uniforme com adição de ruído.

Parâmetro	Valor
Configuração utilizada	2
Distribuição da população inicial	aleatório
Ruído	1; A=3; T=médio; R=0.01
Otimização externa	1 a cada 10 gerações; local=5+gen/10;homo=3
Elitismo	1
Recombinação	1.0 SPAUC; L=2; C=3; F1=0.05; F2=0.05
Parâmetros da rodada	G=150; P=100; PC=0.85; PM=0.03; T=15

Os resultados alcançados foram

Tabela 17 : Resultados da comparação entre SPAUC e Recombinação uniforme com adição de ruído

Rodada	Erro magnético	Erro condutivo
SPAUC 1	0.91	959.9
SPAUC 2	0.59	716.2
SPAUC 3	0.31	262.2
SPAUC 4	0.78	658.7
SPAUC 5	0.39	429.7
SPAUC 6	0.23	262.5
MÉDIA	0.53	548.2
Uniforme 1	0.71	845.5
Uniforme 2	0.56	380.5
Uniforme 3	0.47	748.3
Uniforme 4	0.50	600.6
Uniforme 5	0.84	716.8
Uniforme 6	0.89	915.4
MÉDIA	0.66	701.1

O melhor indivíduo encontrado usando SPAUC o foi na rodada 3 e é

9.9	10.5	9.1	10.9	9.2	11.0	8.8	10.7	10.4	10.0
9.5	8.6	93.8	94.9	90.3	100.0	81.4	10.3	7.2	12.1
11.3	11.9	92.3	100.0	79.8	84.0	97.7	12.6	12.1	11.9
12.9	8.8	6.7	11.3	17.3	43.9	7.6	8.3	8.6	8.9
8.7	9.6	11.9	6.6	8.5	13.3	16.1	11.7	9.5	9.2
10.9	15.9	15.7	7.7	15.0	4.2	7.9	10.1	10.5	16.6
10.0	5.4	14.8	5.4	10.5	7.2	19.5	18.5	8.8	5.8

que pode ser visualizado como

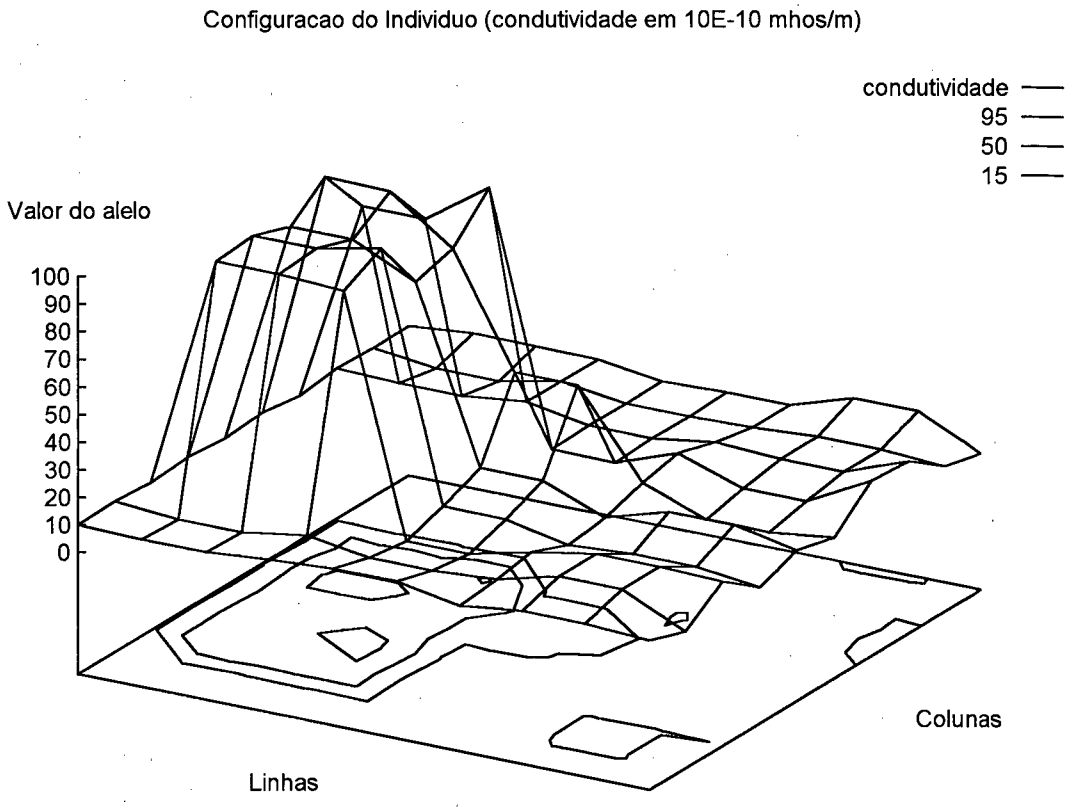


Fig. 19: Melhor indivíduo com SPAUC, dados ruidosos

Já o melhor indivíduo obtido com recombinação uniforme o foi na rodada 2, e é:

9.7	10.5	8.3	12.2	6.8	15.0	6.1	11.7	8.8	11.3
10.7	8.1	92.8	81.2	77.1	100.0	63.4	9.1	11.6	9.2
11.2	10.8	100.0	100.0	86.7	90.1	93.5	11.1	9.1	12.9
9.6	10.5	6.7	22.9	4.4	4.9	80.4	6.3	1.7	32.3
9.2	9.6	13.3	7.1	6.2	25.3	8.2	5.1	15.8	4.8
12.5	11.9	17.8	19.4	10.0	8.6	8.6	8.6	9.0	16.3
7.7	9.0	11.7	4.8	3.2	8.6	8.8	9.0	9.0	9.9

que pode ser visto como

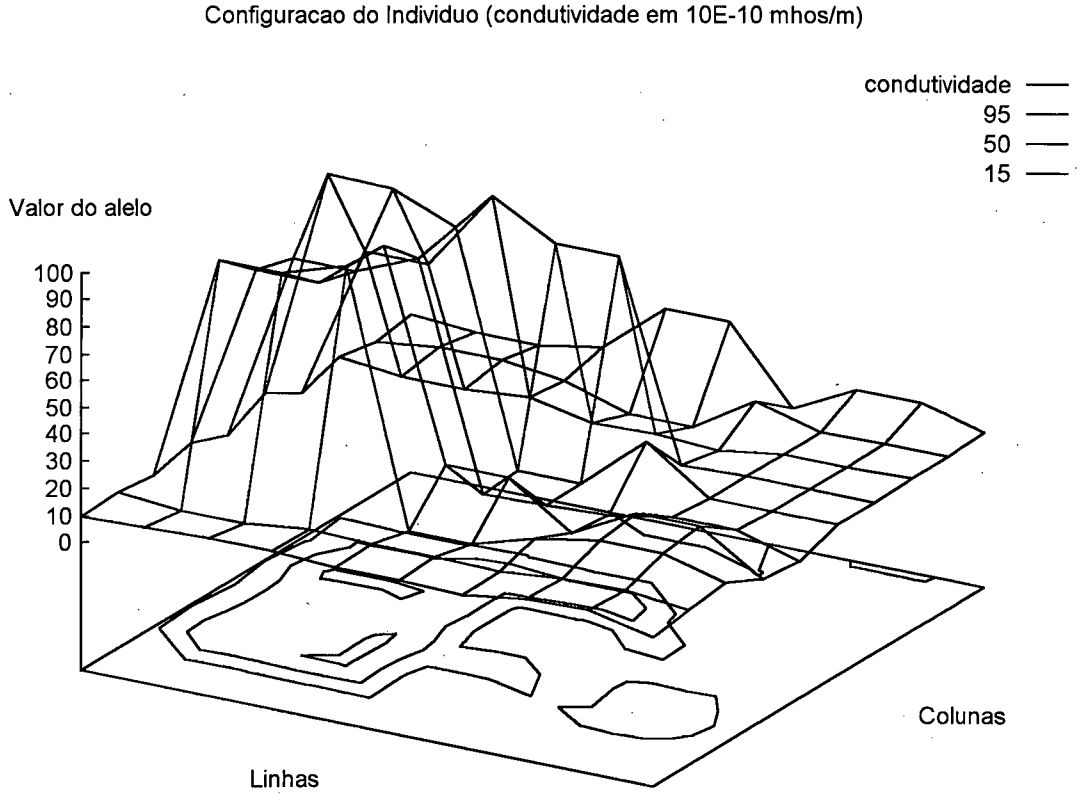


Fig. 20: Melhor indivíduo com recombinação uniforme, dados ruidosos

Pode ser dito que a despeito da inclusão do ruído, SPAUC permanece com melhor desempenho em relação a recombinação uniforme. Na média o erro condutivo do primeiro caso (SPAUC) foi de 548.2 versus 701.1 para o segundo caso.

5.3 Parametrização do Engenho Evolutivo

A seguir apresenta-se uma série de testes efetuados variando-se os principais parâmetros que determinam o funcionamento do modelo. Não se trata de uma busca exaustiva, mas uma variação de cada parâmetro, de maneira independente em relação aos demais, com o objetivo de verificar sua influência na busca por boas soluções para o problema. Observe-se que a suposição de que os parâmetros são independentes uns dos outros é apenas isso: uma suposição.

5.3.1 Regiões de corte em SPAUC

O estudo agora é referente à quantidade de regiões de corte dentro de SPAUC. Relembre-se que quanto mais regiões houverem, mais este operador se aproximará de recombinação uniforme, enquanto que também crescerá o consumo de recursos computacionais, uma vez que a avaliação de cada região implica uma nova chamada à função objetivo.

O objetivo do teste a seguir foi medir o desempenho do operador quando o número de cortes aleatórios é variado. Usou-se a configuração número 1, com 100% dos valores de todos os indivíduos iguais a 10 (x 10^{-11} mhos/m), aplicou-se ruído de desvio padrão igual a 1%, com 3 medidas ruidosas a cada vez, e usando-se o valor médio destas três. A otimização externa ocorreu com ciclo de 10 gerações e a cada ciclo 5 etapas de busca local mais 3 etapas de homogeneização. O número 5 crescia com um décimo do número da geração (por exemplo: na geração 10, ele era chamado 6 vezes = 5 + 10÷10 = 5 + 1; na geração 20, chamado 7 vezes = 5 + 20 ÷ 10 = 5 + 2 = 7, e assim por diante). Usou-se elitismo, a recombinação foi o SPAUC em 100% dos casos. Foram 150 gerações com populações de 100 indivíduos, probabilidade de recombinação igual a 0.85, probabilidade de mutação de 0,03, tamanho de torneio de 15 indivíduos e sementes aleatórias variadas.

A tabela 18 a seguir, resume os parâmetros

Tabela 18 Resumo dos parâmetros na medida de cortes aleatórios em SPAUC

Parâmetro	Valor
Configuração utilizada	1
Distribuição da população inicial	100% igual a 10
Ruído	1; A=3; T=médio; R=0.01
Otimização externa	1 a cada 10 gerações; local=5+gen/10;homo=3
Elitismo	1
Recombinação	1.0 SPAUC; F1=0.05; F2=0.05
Parâmetros da rodada	G=150; P=100; PC=0.85; PM=0.03; T=15; SA=0.333

Os resultados alcançados foram

Tabela 19 : Resultados da comparação para diversos cortes aleatórios em SPAUC		
Rodada	Erro magnético	Erro condutivo
1 corte V e 1 corte H	0.39	148.9
1 V e 1 H	0.64	469.0
1 V e 1 H	1.38	740.9
Média	0.80	452.9
2 V e 1 H	0.71	451.4
2 V e 1 H	0.34	137.1
2 V e 1 H	0.99	487.5
Média	0.68	358.6
2 V e 2 H	0.38	209.9
2 V e 2 H	0.58	314.0
2 V e 2 H	0.73	779.9
Média	0.56	434.6
3 V e 2 H	0.28	150.7
3 V e 2 H	0.19	79.5
3 V e 2 H	0.41	146.3
Média	0.29	125.5

Os resultados são crescentemente melhores à medida em que o número de regiões de SPAUC aumenta, levando-se em conta que após a realização da recombinação, as regiões formadas vão (poder) sofrer uma homogeneização. Este parênteses é importante, pois se levada à última conseqüência, este teste indicaria que a recombinação uniforme seria melhor que SPAUC, o que contradiz o teste anterior. Deve-se ressaltar também que quanto maior o número de regiões em SPAUC maior o consumo de recursos o que atenua a tendência de usar muitas regiões no operador.

5.3.2 Homogeneização em filhos de SPAUC

Este teste teve como objetivo verificar a importância da homogeneização que é feita logo após a operação de recombinação. Usou-se a configuração número 1, com 100% dos valores de todos os indivíduos iguais a 10 (x 10^{-11} mhos/m), aplicou-se ruído de desvio padrão igual a 1%, com 3 medidas ruidosas a cada vez, e usando-se o valor médio destas três. A otimização externa ocorreu com ciclo de 10 gerações e a cada ciclo 5 etapas de busca local mais 3 etapas de homogeneização. O número 5 crescia com um décimo do número da geração. Usou-se elitismo, a recombinação foi o SPAUC em 100% dos casos. Foram 150 gerações com populações de 100 indivíduos, probabilidade de recombinação igual a 0.85, probabilidade de mutação de 0,03, tamanho de torneio

de 15 indivíduos e sementes aleatórias variadas. A tabela 20 resume os parâmetros utilizados no teste.

Tabela 20 : Parâmetros usados no teste de probabilidade de homogeneização dos descendentes após SPAUC

Parâmetro	Valor
Configuração utilizada	1
Distribuição da população inicial	100% igual a 10
Ruído	1; A=3; T=médio; R=0.01
Otimização externa	1 a cada 10 gerações; local=5+gen/10;homo=3
Elitismo	1
Recombinação	1.0 SPAUC; L=2; C=3
Parâmetros da rodada	G=150; P=100; PC=0.85; PM=0.03; T=15; SA=0.333

Os resultados alcançados foram

Tabela 21 : Resultados da comparação para as probabilidades PH de homogeneização dos descendentes após SPAUC

Rodada	Erro magnético	Erro condutivo
sem homogeneização, PH = 0.0	0.48	267.4
PH = 0.0	0.61	262.1
PH = 0.0	1.37	693.3
Média	0.82	407.6
PH = 0.025	0.45	302.8
PH = 0.025	0.78	645.4
PH = 0.025	0.56	424.0
Média	0.59	457.4
PH = 0.05	0.28	150.7
PH = 0.05	0.19	79.5
PH = 0.05	0.41	146.3
Média	0.29	125.5

Quanto maior a probabilidade de uma dada região que foi recombinada usando SPAUC, sofrer homogeneização, melhor é o resultado alcançado. Novamente atuam duas tendências irreconciliáveis: enquanto maior probabilidade de homogeneização indica resultados melhores, ela também aumenta o consumo de recursos. A solução de compromisso aqui (como em outros locais) é usar a configuração menos custosa que gera resultados satisfatórios.

5.3.3 Busca Local

Uma busca local é um operador que atua no sentido contrário ao da homogeneização. Ela privilegia descontinuidades, forçando-as a aparecer. Trata-se de um algoritmo simples de subida da encosta, que atua buscando melhorar o valor atual pela pesquisa de direções de melhoria no entorno do ponto inicial. No dizer de Michalewicz (1997), trata-se de uma *mutação inteligente.* Da maneira como foi programado, este operador exige dois parâmetros: em qual período de gerações ele atuará e com quantos ciclos ele será usado cada vez que for chamado. Novamente, tem-se aqui uma busca de equilíbrio. Quanto menor o período e quanto maior o ciclo, mais efetiva será a busca local. Em compensação, mais recursos serão demandados, uma vez que a cada aplicação individual (um único valor e um único incremento) há que se chamar a função objetivo.

Na maioria das aplicações deste operador neste trabalho, usou-se um período de 10 gerações. Como quase todas as rodadas foram de 150 gerações, isto significou cerca de 15 buscas locais por rodada. Já a quantidade de ciclos por aplicação foi estabelecida como variável. A justificativa dessa opção é que o engenho evolutivo atua muito fortemente no início não sendo muito importante a busca local nesta situação. À medida em que o processo evolui, deixa de haver saltos bruscos e a pesquisa passa a ser mais refinada. Nesta hora a busca local tem o poder de otimizar bastante o resultado. Na maioria das aplicações, a fórmula que deu o número de ciclos foi de

$$ciclos = 5 + \text{int}(\frac{geracao}{10})$$

A cada ciclo, são feitas 140 tentativas de melhoria do resultado. O número 140 se explica porque o cromossomo é formado por 70 números e cada um deles têm um pequeno incremento (randômico) positivo e outro negativo para ver se o resultado é melhorado. As posições que melhoram o resultado são guardadas e ao final do ciclo todas são aplicadas ao mesmo tempo. O indivíduo resultante substitui o original de fitness menor.

Acompanhe a aplicação no exemplo:

Seja um cromossomo qualquer

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24

que como se sabe tem um fitness F_1.

Um ciclo, iniciaria pela geração de um número randômico tipicamente infinitesimal. Este número é somado a cada célula do cromossomo, por exemplo

1+δ	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24

Após esta soma, a função objetivo é recalculada, tendo como resultado F_1'. Se F_1' é maior que F_1, guarda-se a informação +1 em um *array* similar ao cromossomo, e na posição equivalente.

+1					

Caso não tenha havido melhoria na função objetivo, repete-se o teste agora usando o mesmo infinitésimo com valor negativo, por exemplo

1-δ	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24

Novamente a função objetivo é recalculada, gerando um fitness igual a F_1''. Caso F_1'' seja maior que F_1, guarda-se a informação -1 no *array* similar de controle.

Isso se repete para todos os números, e ao final todos os incrementos e decrementos são aplicados em uma só vez, observados os limites numéricos de cada alelo, para impedir estouros positivos e negativos *(under e overflow)*. Note-se que este operador considera que os componentes do cromossomo são independentes uns dos outros (Se a_{11} força o crescimento do fitness e a_{12} idem, ambos devem ser aumentados. Mas, e

é um mas importante, pode ser que em uma dada função, ainda que individualmente ambos aumentem o fitness, o crescimento conjunto dos dois valores force uma baixa no fitness). É mais uma manifestação do fenômeno conhecido como epistasia. Aqui não foi possível detectar a presença ou ausência de manifestações epistáticas. De qualquer maneira, é bom lembrar que a busca local atua apenas como auxiliar ao engenho evolutivo, que é quem de fato orienta a busca.

O uso da busca local não é uma abordagem inédita. Bäumer (1996), estudando um problema similar diz textualmente "a maneira mais eficiente é iniciar com um algoritmo genético para obter uma população próxima do ótimo e então usar esta população como ponto inicial para processos de busca local" .

O valor de δ que é utilizado para forçar o salto na busca local é ainda multiplicado pelo índice i que indica qual a profundidade do valor para o qual se verifica se o salto implicará melhora no *fitness*. Assim, para a primeira linha, o multiplicador é 1, para a segunda 2, para a terceira 3 e assim por diante. A última linha tem profundidade 7 o que implica que os valores de δ são multiplicados por 7. A justificativa para este procedimento está na sensibilidade menor que o modelo apresenta, à medida em que a profundidade aumenta, conforme pode ser comprovado olhando a tabela 12.

O teste a seguir busca avaliar qual o mínimo de chamadas que permite que a busca local faça sentir seus efeitos e a importância da homogeneização que é feita logo após a operação de recombinação. Usou-se a configuração número 1, com 100% dos valores de todos os indivíduos iguais a 10 (x 10^{-11} mhos/m), aplicou-se ruído de desvio padrão igual a 1%, com 3 medidas ruidosas a cada vez, e usando-se o valor médio destas três. Usou-se elitismo, a recombinação foi o SPAUC em 100% dos casos. Foram 150 gerações com populações de 100 indivíduos, probabilidade de recombinação igual a 0.85, probabilidade de mutação de 0,03, tamanho de torneio de 15 indivíduos e sementes aleatórias variadas. A tabela 22 resume os parâmetros utilizados no teste.

Tabela 22: Resumo dos parâmetros de teste da quantidade de chamadas à busca local

Parâmetro	Valor
Configuração utilizada	1
Distribuição da população inicial	100% igual a 10
Ruído	1; A=3; T=médio; R=0.01
Otimização externa	1 a cada 10 gerações; homo=3
Elitismo	1
Recombinação	1.0 SPAUC; L=2; C=3; F1=0.05; F2=0.05
Parâmetros da rodada	G=150; P=100; PC=0.85; PM=0.03; T=15; SA=0.333

Os resultados a que se chegou, foram

Tabela 23: Resultados da comparação para as quantidades de chamadas à busca local		
Rodada	Erro magnético	Erro condutivo
10 etapas por ciclo	0.27	103.5
10	1.25	571.8
10	0.48	224.3
Média	0.66	299.8
5	0.30	117.4
5	1.25	686.4
5	0.82	381.6
Média	0.79	395.1
5 + G/10	0.28	150.7
5 + G/10	0.19	79.5
5 + G/10	0.41	146.3
Média	0.29	125.5

O caso 1, que corresponde a 10 chamadas é superior ao caso 2 (5 chamadas) como era de se imaginar. Entretanto o caso 1 demanda maiores consumos de máquina, pois cada chamada a mais implica 140 invocações da função objetivo. No total, a diferença de consumo entre o caso 1 e o caso 2 é de 700 chamadas. Tal consumo é desnecessário no início, sendo entretanto adequado ao final do processo. O caso 3, parece resumir satisfatoriamente o compromisso adotado: poucas chamadas no início e maior número de chamadas ao final. Veja-se na tabela 24, o a quantidade de chamadas por geração.

Tabela 24: Composição do número de chamadas à função objetivo a cada geração															
Geração	10	20	30	40	50	60	70	80	90	100	110	120	130	140	150
Número de Chamadas por ciclo	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

O código fonte comentado da busca local se encontra no anexo B.

5.3.4 Homogeneização

Este operador se baseia no conceito de que regiões vizinhas terão probabilidade alta de terem condutividades iguais. Cada tentativa de aplicação deste operador se inicia pela geração de uma região retangular randômica (regiões pequenas têm alta probabilidade de geração. Regiões grandes têm baixa probabilidade). Todos os valores da região são substituídos a cada vez por cada um dos elementos que fazem parte da região. Valores que ao serem distribuídos na região melhoram a função objetivo são guar-

dados. Ao final, aquela modificação que gerou a maior melhora do resultado (se é que houve algum) é aplicada a toda a região.

Por exemplo, suponhamos um determinado cromossomo

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24

que tem um determinado fitness, por exemplo F_1. A atuação do operador de homogeneização, inicia pela escolha de uma região aleatória, por exemplo

1	2	3	4	5	6
7	8	[9	[10	11	12
13	14	[15	[16	17	18
19	20	[21	[22	23	24

Agora, o cromossomo terá os valores da região sombreada substituídos pelos valores 9, 10, 15, 16, 21 e 22 a cada vez. A primeira substituição será

1	2	3	4	5	6
7	8	[9	[9	11	12
13	14	[9	[9	17	18
19	20	[9	[9	23	24

Este novo cromossomo terá valor F_1'. Este valor será guardado. A próxima substituição é

1	2	3	4	5	6
7	8	[10	[10	11	12
13	14	[10	[10	17	18
19	20	[10	[10	23	24

que dará origem a um novo valor F_1'', que também será guardado, e assim por diante até que todos os valores do bloco tenham sido testados.

Ao final, serão 6 valores de fitness. No caso mais comum, nenhum deles será maior que F1, e portanto este operador nenhuma melhoria trouxe para o processo. Entretanto, há situações em que algum dos valores é maior que F_1. Se isto ocorrer, o cro-

mossomo modificado que deu origem ao valor maior de fitness, substitui o cromossomo original. Se mais de um valor de fitness é maior do que original, usa-se o maior valor.

Note-se que este operador é um grande consumidor de recursos, pois para cada um dos valores que compõe a região escolhida para sofrer o operador, há que se fazer uma chamada à função objetivo. Por esta razão, e pelo fato de que pequenas regiões têm maior potencial a permitirem um aumento do fitness através da homogeneização, do que as regiões grandes, este operador privilegia regiões pequenas.

O algoritmo que garante probabilidades maiores para regiões menores se encontra a seguir:

```
inteiro função fg
flutuante n, sum
n = 0
sum = 0
bi_gauss = [***probabilidades desejadas ***]
int i = 0
n = aleatorio();  [gera-se um número aleatório entre 0..1]
enquanto (sum ≤ n) e (i < ***quantidade de probabilidades ***)
     sum = sum + bi_gauss [i]
     i++
fim-enquanto
devolva i
fim-função
```

Para a obtenção das probabilidades desejadas, usou-se o seguinte argumento: o maior número de linhas (ou colunas) possível para o bloco teria associado a ele o número natural 1. O tamanho imediatamente menor estaria associado a 2 (ou seja com o dobro de probabilidade de ser escolhido). O seguinte, associado a 3 e assim sucessivamente. Para os cortes horizontais, definiu-se que o maior bloco poderia ter 7 linhas. O bloco de 6 linhas teria o dobro de probabilidade, o de 5 linhas o triplo e até o bloco de 2 linhas que tem 6 vezes mais probabilidade de ser gerado.

Em resumo, para as linhas eis as probabilidades

Tabela 25 Probabilidades de tamanhos de blocos na homogeneização, para as colunas

tamanho do bloco (em linhas)	probabilidades (como múltiplos da menor delas)	probabilidade
2	6	6/21 = 0.2857
3	5	5/21 = 0.2380
4	4	4/21 = 0.1904
5	3	3/21 = 0.1428
6	2	2/21 = 0.0952

7	1	1/21 = 0.0476
	Soma = 21	Soma = 1.0

Portanto, as "probabilidades desejadas" citadas no código acima, são 0.2857, 0.2380, 0.1904, 0.1428, 0.0952 e 0.0476.

De maneira idêntica, para as colunas foram usados os seguintes resultados:

Tabela 26 - Probabilidades de tamanhos de blocos na homogeneização, para as linhas

tamanho do bloco (em colunas)	probabilidades (como múltiplos da menor delas)	probabilidade
2	8	8/36 = 0.2222
3	7	7/36 = 0.1944
4	6	6/36 = 0.1666
5	5	5/36 = 0.1388
6	4	4/36 = 0.1111
7	3	3/36 = 0.0833
8	2	2/36 = 0.0555
9	1	1/36 = 0.0277
	Soma = 36	Soma = 1.0

Os testes a seguir, descrevem os diversos comportamentos obtidos pelo modelo quando se variou a quantidade de homogeneizações. Usou-se a configuração número 1, com 100% dos valores de todos os indivíduos iguais a 10 (x 10^{-11} mhos/m), aplicou-se ruído de desvio padrão igual a 1%, com 3 medidas ruidosas a cada vez, e usando-se o valor médio destas três. A otimização externa ocorreu com ciclo de 10 gerações e a cada ciclo 5 etapas de busca local. Usaram-se 2 cortes horizontais e 3 verticais para o SPAUC. Usou-se elitismo, a recombinação foi o SPAUC em 100% dos casos. Foram 150 gerações com populações de 100 indivíduos, probabilidade de crossover igual a 0.85, probabilidade de mutação de 0,03, tamanho de torneio de 15 indivíduos e sementes aleatórias variadas. A tabela 27 indica os parâmetros usados

Tabela 27 - Parâmetros usados no teste de variação da quantidade de homogeneizações

Parâmetro	Valor
Configuração utilizada	1
Distribuição da população inicial	100% igual a 10
Ruído	1; A=3; T=médio; R=0.01
Otimização externa	1 a cada 10 gerações; local=5+gen/10
Elitismo	1
Crossover	1.0 SPAUC; L=2; C=3; F1=0.05; F2=0.05

Parâmetros da rodada	G=150; P=100; PC=0.85; PM=0.03; T=15; SA=0.333

Os resultados foram

Tabela 28 : Resultados da comparação para as quantidades de homogeneizações

Rodada	Erro magnético	Erro condutivo
0 homogeneizações	0.28	130.9
0	0.92	712.1
0	1.24	657.1
Média	0.81	500.0
1	0.35	195.9
1	0.82	708.6
1	1.17	649.3
Média	0.78	517.9
2	0.29	133.6
2	0.97	801.4
2	1.42	620.2
Média	0.89	518.4
3	0.28	150.7
3	0.19	79.5
3	0.41	146.3
Média	0.29	125.5

Como conclusão desta verificação, percebe-se melhoria crescente à medida em que cresce o número de homogeneizações realizadas a cada ciclo. O único resultado destoante ocorre no caso 2 e se deve a um resultado atípico (1.42 de erro magnético). Desprezando-se este caso o conjunto restante é coerente com a melhoria proporcional ao número maior de homogeneizações.

O código fonte da homogeneização se encontra no apêndice E.

5.3.5 Influência da inicialização da população

Nos testes a seguir, buscou-se determinar qual a influência que a inicialização da população com um determinado valor constante (e igual à rocha do entorno) causaria no resultado final.

Usou-se a configuração 1, com 1% de ruído gaussiano, 3 medidas distintas, ficando-se com a média entre as 3 medidas. Usou-se um ciclo de busca local e homogeneização a cada 10 gerações e cada ciclo era composto por 5+geração / 10 etapas de busca local e 3 etapas de homogeneização. Usou-se elitismo e o operador de recombinação é o SPAUC em 100% dos casos com 2 cortes horizontais e 3 verticais (12 regiões), com probabilidade de homogeneização dos descendentes igual a 5%. Foram 150

gerações, com 100 indivíduos na população, probabilidades de crossover e mutação de 85% e 3% respectivamente, com tamanho de torneio de 15 indivíduos.

A tabela 29 resume estes parâmetros.

Tabela 29 : Parâmetros no teste de inicialização da população

Parâmetro	Valor
Configuração utilizada	1
Ruído	1; A=3; T=médio; R=0.01
Otimização externa	1 a cada 10 gerações; local=5+gen/10;homo=3
Elitismo	1
Crossover	1.0 SPAUC; L=2; C=3; F1=0.05; F2=0.05
Parâmetros da rodada	G=150; P=100; PC=0.85; PM=0.03; T=15; SA=0.333

Os resultados obtidos foram

Tabela 30: Resultados da comparação para a influência da inicialização

Rodada	Erro magnético	Erro condutivo
0 % de valores iguais a 10	1.14	545.3
0 %	0.75	433.7
0 %	0.53	288.3
Média	**0.80**	**422.4**
30 % igual a 10	0.45	177.9
30%	0.40	289.2
Média	**0.42**	**233.5**
60% igual a 10	0.80	348.0
60%	0.35	118.4
60%	0.38	175.8
Média	**0.51**	**214.0**
100% igual 10	0.28	150.7
100%	0.19	79.5
100%	0.41	146.3
Média	**0.29**	**125.5**

Como se utilizou a configuração 1 que tem um grande número de prismas cuja condutividade é 10 (x 10^{-11} mhos/m), este valor foi considerado como sendo o valor de contorno. Percebe-se que quanto mais próximos deste valor forem os indivíduos da população inicial, menores são os erros condutivo e magnético e conseqüentemente mais correto é o resultado. A dificuldade maior aqui, para um caso real, é determinar qual seja este valor de contorno.

5.3.6 Periodicidade a chamadas a operadores ad-hoc

Nos testes a seguir, verificou-se buscar qual o desempenho ideal em termos de chamadas aos ciclos de busca local e homogeneização. Usou-se a configuração 1, com 1% de ruído gaussiano, 3 medidas distintas, ficando-se com a média entre as 3 medidas. Testava-sa qual a configuração ideal para o número de ciclos e cada ciclo era composto por 5+geração / 10 etapas de busca local e 3 etapas de homogeneização. 100% dos indivíduos era inicializado com o valor 10. Usou-se elitismo e o operador de recombinação é o SPAUC em 100% dos casos com 2 cortes horizontais e 3 verticais (12 regiões), com probabilidade de homogeneização dos descendentes igual a 5%. Foram 150 gerações, com 100 indivíduos na população, probabilidades de crossover e mutação de 85% e 3% respectivamente, com tamanho de torneio de 15 indivíduos.

A tabela 31, resume estes parâmetros.

Tabela 31 : Resumo dos parâmetros de quantidade de ciclos de otimização	
Parâmetro	Valor
Configuração utilizada	1
Distribuição da população inicial	100% igual a 10
Ruído	1; A=3; T=médio; R=0.01
Elitismo	1
Crossover	1.0 SPAUC; L=2; C=3; F1=0.05; F2=0.05
Parâmetros da rodada	G=150; P=100; PC=0.85; PM=0.03; T=15; SA=0.333

Os resultados estão na tabela a seguir:

Tabela 32 : Resultados da comparação para a quantidade de ciclos de otimização		
Rodada	Erro magnético	Erro condutivo
1 ciclo a cada 40 gerações	0.42	113.5
40	1.82	631.6
40	1.38	388.5
Média	1.20	377.8
1 ciclo a cada 30 gerações	0.36	140.7
30	1.29	619.8
30	1.24	513.6
Média	0.96	424.7
1 ciclo a cada 20 gerações	0.46	210.6
20	1.33	600.3
20	1.30	526.7
Média	1.03	445.8
1 ciclo a cada 10 gerações	0.28	150.7

10	0.19	79.5
10	0.41	146.3
Média	0.29	125.5

Embora não haja uma homogeneidade na tendência à queda dos erros condutivo e magnético à medida em que diminui a quantidade de gerações entre os ciclos de otimização, pode-se observar uma tendência à queda destes erros. Considerando que se estabeleceu uma quantidade de 150 gerações por execução, para o caso 1, teve-se 3 ciclos (nas gerações 40, 80 e 120). O caso 2, teve aplicação nas gerações (30, 60, 90, 120 e 150). O caso 3, nas gerações 20, 40, 60, 80, 100, 120 e 140 e o caso 4 em 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140 e 150. Totalizando-se, verifica-se que a série consistiu de 3, 5, 7 e 15 aplicações de ciclos de otimização. Talvez a irregularidade da série 3, 5, 7 e 15 ajude a explicar a irregularidade no resultado dos testes. É uma hipótese. Entretanto, não há como deixar de notar que quanto maior a aplicação de ciclos de otimização, menores são os erros.

5.3.7 Aplicação de Ruído

Os testes a seguir buscam determinar a influência das maneiras de calcular a aplicação de ruído gaussiano sobre a medida. O processo se iniciava pela aplicação de ruído à medida original de condutividade em três rodadas distintas. Como a aplicação de ruído é processo estocástico, do qual apenas se conhece o desvio padrão da distribuição, certamente três distribuições distintas resultavam. O resultado da aplicação da função objetivo ao candidato a solução era em seguir comparado com as três medidas originais. O resultado final, poderia ser: a) a melhor das três comparações; b) a pior delas e c) a média entre as três. É isso que se estudou aqui. Dizendo de outra forma, a hipótese era de que se se trabalhasse sempre com o melhor (menor resultado) das comparações do campo elétrico nem sempre se teria o melhor resultado em termos de condutividade. Usou-se a configuração 1, com 1% de ruído gaussiano, 3 medidas distintas, e o processamento desta medida era o objeto do teste. O ciclo de otimização era chamado a cada 10 gerações e cada ciclo era composto por 5+geração / 10 etapas de busca local e 3 etapas de homogeneização. 100% dos indivíduos era inicializado com o valor 10. Usou-se elitismo e o operador de recombinação é o SPAUC em 100% dos casos com 2 cortes horizontais e 3 verticais (12 regiões), com probabilidade de homogeneização dos descendentes igual a 5%. Foram 150 gerações, com 100 indivíduos na população, pro-

babilidades de crossover e mutação de 85% e 3% respectivamente, com tamanho de torneio de 15 indivíduos.

A tabela 33 resume isto

Tabela 33 - Resumo dos parâmetros do teste de processamento de ruído	
Parâmetro	Valor
Configuração utilizada	1
Distribuição da população inicial	100% igual a 10
Ruído	1; A=3; R=0.01
Otimização externa	1 a cada 10 gerações; local=5+gen/10;homo=3
Elitismo	1
Crossover	1.0 SPAUC; L=2; C=3; F1=0.05; F2=0.05
Parâmetros da rodada	G=150; P=100; PC=0.85; PM=0.03; T=15; SA=0.333

Os resultados foram os seguintes:

Tabela 34 - Resultados do processamento de ruído		
Rodada	Erro magnético	Erro condutivo
pior resultado	0.32	85.1
pior	1.38	669.9
pior	1.44	574.2
Média	1.04	443.0
melhor resultado	0.48	157.8
melhor	0.91	689.5
melhor	0.46	308.2
Média	0.61	385.1
resultado médio	0.28	150.7
médio	0.19	79.5
médio	0.41	146.3
Média	0.29	125.5

O resultado melhor, apresentava – como de resto era esperado – resultados numéricos melhores na aplicação pura e simples da função objetivo. Tipicamente o piso aqui era de 1.2965. Por idêntica razão, a aplicação do resultado pior afastava-se desse mesmo valor. Um resultado típico seria 1.4869. Na média, o valor piso era de 1.4408. Entretanto, essa diferença não se transferiu -- novamente como era esperado -- para os resultados dos erros, e a escolha da média, parece estar justificada como um meio termo entre diferenças de campo elétrico e condutividade.

5.4 Comparação entre um método evolutivo específico do problema e uma abordagem clássica:

A seguir descrevem-se testes efetuados comparando-se os resultados de rodadas evolutivas versus rodadas similares usando-se a abordagem clássica adicionada de

regularização. Note-se que a dupla de parâmetros de regularização γ_0 e γ_1, não têm o mesmo valor para todos os casos. Ao contrário, na ausência de uma lei de formação para os valores ideais de gamma0 e gamma1, a cada caso, valores distintos são usados. A rigor, em cada um, usou-se o par que melhor desempenho apresentou.

5.4.1 Caso 1 - Rodadas com otimização via gradiente mais regularização

Neste teste, comparam-se diversas combinações de γ_0 e γ_1, na abordagem clássica. Usou-se a configuração 1, com 100% dos valores da solução inicial igual a 10. A aplicação de ruído é de 1%. A tabela 35 resume os parâmetros.

Tabela 35: Testes da abordagem clássica

Parâmetro	Valor
Configuração utilizada	1
Distribuição da população inicial	100% igual ao valor 10
Ruído	0.01
Parâmetros da rodada	γ_0 = 0.01 e γ_1 = 0.0

Eis os resultados

Tabela 36 : Resultados da variação de γ_0 e γ_1 (abordagem clássica)

Rodada	Erro magnético	Erro condutivo
0.01 e 0.0	6.08	114.8
0.02 e 0.0	8.95	593.5
0.03 e 0.0	11.38	387.1
0.04 e 0.0	8.05	500.1
0.05 e 0.0	6.01	94.0
0.0 e 0.01	6.00	29.5
0.0 e 0.02	9.88	27.1
0.0 e 0.03	10.00	176.6
0.0 e 0.04	7.21	200.4
0.0 e 0.05	10.09	193.8
não conhecido	0.59	382.11
não conhecido	0.40	150.0
não conhecido	0.51	304.4
não conhecido	0.13	7.3
não conhecido	0.84	449.0

O melhor indivíduo é

```
 10.1   9.9  10.1   9.9   9.8   9.7  10.1  10.0  10.0  10.0
 10.0   9.9   1.0   1.0   9.6 100.0 100.0  10.0  10.0  10.0
 10.0   9.9   1.0   1.1   9.6 100.0 100.0  10.0  10.0  10.0
  9.8   9.9  10.0  10.0   9.6  10.1   9.6  10.0  10.0  10.0
  9.8   9.9  10.0  10.0   9.6  10.1   9.6  10.0  10.0  10.0
```

```
9.9  10.1  10.0  10.0   9.6  10.1   9.6  10.0  10.0  10.0
9.9  10.1   9.9   9.8   9.6  10.1   9.6  10.0  10.0  10.0
```

cuja representação é

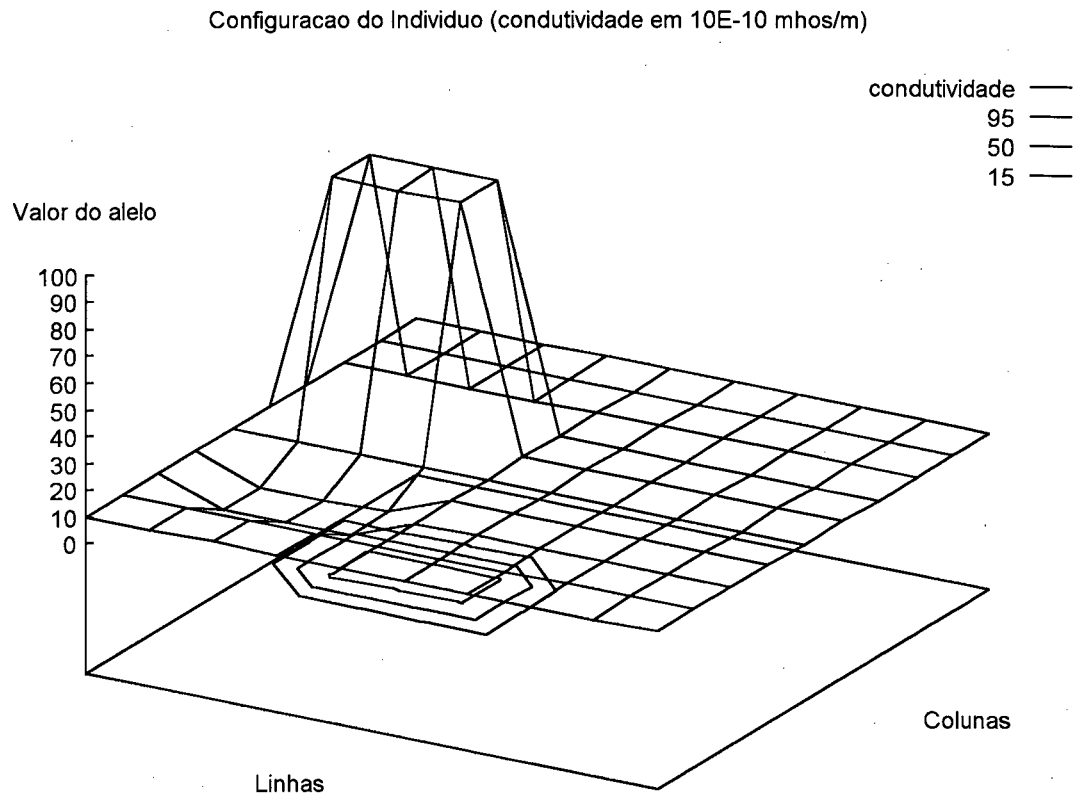


Fig. 21: Melhor resultado para o caso 1 em (Ramos e Campos Velho, 1996)

As diversas execuções do modelo clássico são consistentes entre si e todas apontam para uma solução do problema, ainda que varie bastante o resultado alcançado. Há duas classes distintas no que se refere ao erro magnético. Essa separação desaparece quando se analisa o erro condutivo, que é o árbitro final do resultado. O melhor resultado alcançado, quando se usou $\gamma_0 = 0.00$ e $\gamma_1 = 0.02$ é um excelente resultado, conforme pode-se ver na sua representação tridimensional, sendo o melhor resultado obtido em todo o estudo.

5.4.2 Caso 1 - Rodadas do Engenho Evolutivo

Visam os testes a seguir a comparação entre a abordagem convencional e a evolutiva. Os testes usaram a configuração 1, com 1% de ruído gaussiano, 3 medidas distintas, com o valor médio entre as três. O ciclo de otimização era chamado a cada 10 gerações e cada ciclo era composto por 5+geração / 10 etapas de busca local e 3 etapas de homogeneização. 100% dos indivíduos era inicializado com o valor 10. Usou-se elitismo e o operador de recombinação é o SPAUC em 100% dos casos com 2 cortes horizontais e 3 verticais (12 regiões), com probabilidade de homogeneização dos descendentes igual a 5%. Foram 150 gerações, com 100 indivíduos na população, probabilidades de crossover e mutação de 85% e 3% respectivamente, com tamanho de torneio de 15 indivíduos. A tabela 37 resume os parâmetros.

Tabela 37 - Testes da configuração evolutiva

Parâmetro	Valor
Configuração utilizada	1
Distribuição da população inicial	100% igual a 10
Ruído	1; A=3; T=médio; R=0.01
Otimização externa	1 a cada 10 gerações; local=5+gen/10;homo=3
Elitismo	1
Crossover	1.0 SPAUC; L=2; C=3; F1=0.05; F2=0.05
Parâmetros da rodada	G=150; P=100; PC=0.85; PM=0.03; T=15

Os resultados

Tabela 38 - Resultados do engenho evolutivo

Rodada	Erro magnético	Erro condutivo
1	0.19	79.5
2	0.41	146.3
3	0.29	172.2
4	0.50	276.6
5	0.46	197.9
6	0.40	114.8
7	0.52	182.6
8	0.41	117.6
9	0.52	275.6
10	0.37	126.1
11	0.41	218.5
12	0.29	131.1
13	0.31	275.0
Média	0.39	177.9

O melhor indivíduo, obtido na rodada 1

```
 9.9   10.0    9.7   10.1    9.8   10.2    9.7   10.2   10.5    9.6
 9.2   11.0    1.0    1.0   10.9   95.5   97.2   10.0    9.4   11.2
```

11.2	8.7	1.1	1.2	8.5	97.3	97.9	10.1	9.4	11.2
11.5	10.1	10.1	9.2	9.9	9.7	11.0	11.4	10.6	10.2
8.4	10.9	9.9	10.1	12.1	10.0	11.9	10.5	10.1	8.3
12.5	13.2	10.1	8.0	8.8	9.5	6.6	8.9	11.8	13.4
7.1	8.9	10.5	11.8	9.4	7.2	12.0	13.8	9.8	5.6

cuja representação é:

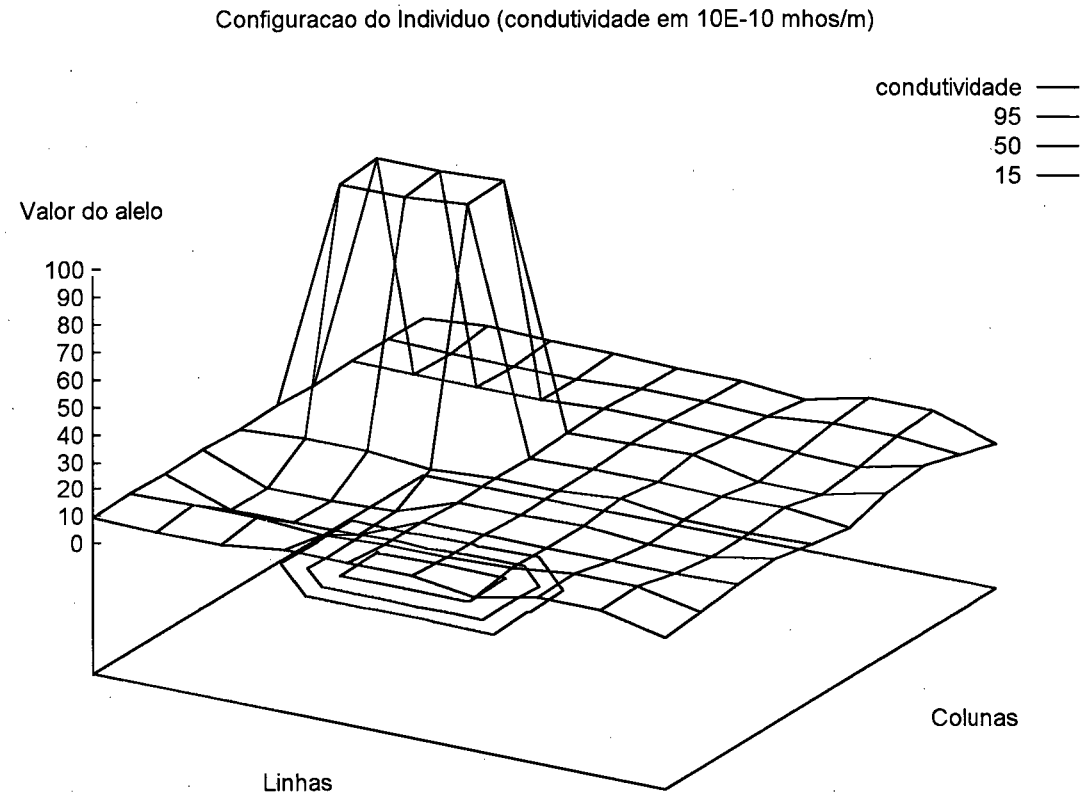


Fig. 22: Melhor resultado para o caso 1 usando Computação Evolutiva

O engenho evolutivo é robusto. O maior erro condutivo (ou seja, o indivíduo mais distante do objetivo) é aproximadamente 270 (x 10^{-11} mhos/m), o que é pouco comparado com o maior erro da abordagem clássica com regularização que é cerca de 590 (x 10^{-11} mhos/m). Este fato se contrapõe ao fato de que o melhor indivíduo aqui é pior do que na abordagem clássica. A simples inspeção visual confirma o fato, além de que o erro condutivo é superior ao dobro: de 27.1 passou para 79.5.

5.4.3 Níveis diferenciados de ruído - rodadas com otimizador via gradiente mais regularização

Os testes agora visam determinar a imunidade a ruído. Inicia-se pela abordagem clássica. A configuração buscada é a 1, os valores são todos inicializados com o valor da rocha, e os valores de χ_0 e χ_1 são 0.0 e 0.01 respectivamente. A tabela 39 resume isso.

Tabela 39 Parâmetros dos testes de imunidade a ruído, abordagem clássica

Parâmetro	Valor
Configuração utilizada	1
Distribuição da população inicial	100% igual ao valor 10
Ruído	0.02, 0.03, 0.04 e 0.08 respectivamente
Parâmetros da rodada	$\gamma_0 = 0.0$ e $\gamma_1 = 0.01$

Os resultados

Tabela 40 Resultados da imunidade ao ruído

Rodada	Erro magnético	Erro condutivo
2%	6.31	198.5
3%	2.89	430.9
4%	9.07	752.5
8%	13.74	1051.3

Veja-se graficamente a degradação ao ruído

Melhor indivíduo com erro gaussiano de 2%

9.8	9.3	9.8	9.6	8.7	9.1	10.6	10.0	9.9	10.6
9.8	10.6	1.4	1.7	8.7	100.0	100.0	10.0	9.9	9.0
9.2	10.6	1.0	1.1	11.9	31.7	37.3	10.0	10.0	9.1
9.2	10.6	5.1	1.6	11.9	10.6	8.2	10.0	10.6	9.1
9.3	10.7	9.5	8.7	11.9	10.6	8.2	10.0	10.7	6.8
9.3	10.7	9.6	8.7	11.9	10.6	8.2	9.9	10.6	6.8
9.3	10.7	9.6	8.7	9.1	10.6	8.2	10.0	10.6	6.8

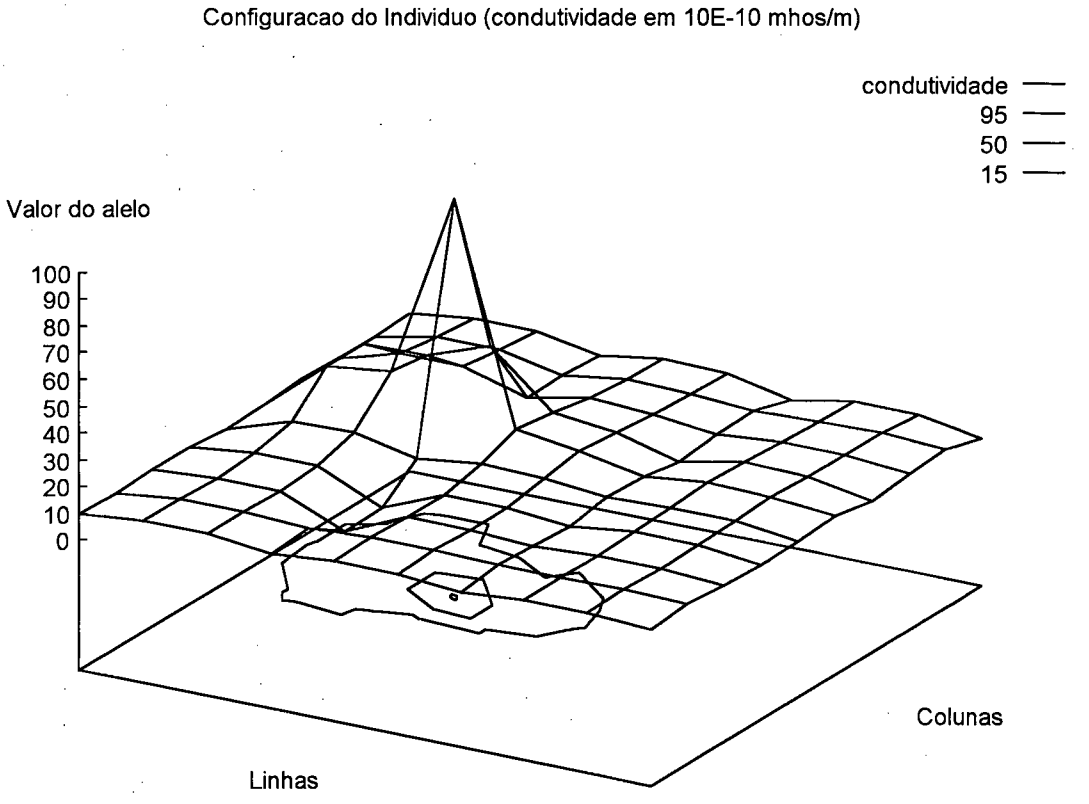


Fig. 23: Rodada clássica: ruído de 2%

Melhor indivíduo com erro gaussiano de 3%

9.7	7.9	8.7	10.7	8.2	8.7	11.0	11.3	9.0	12.1
9.7	16.6	1.8	1.4	8.2	100.0	100.0	6.5	9.2	9.4
8.2	6.8	1.5	1.9	10.7	100.0	100.0	17.2	10.1	7.2
4.6	6.9	1.8	1.8	11.0	10.3	4.6	17.2	10.6	8.0
7.3	26.7	7.2	19.2	2.8	10.3	4.4	5.5	10.5	8.4
6.6	9.0	.3	52.6	45.4	6.8	3.7	85.4	18.5	.9
6.1	8.5	15.1	52.9	.7	1.2	1.6	7.6	18.5	1.6

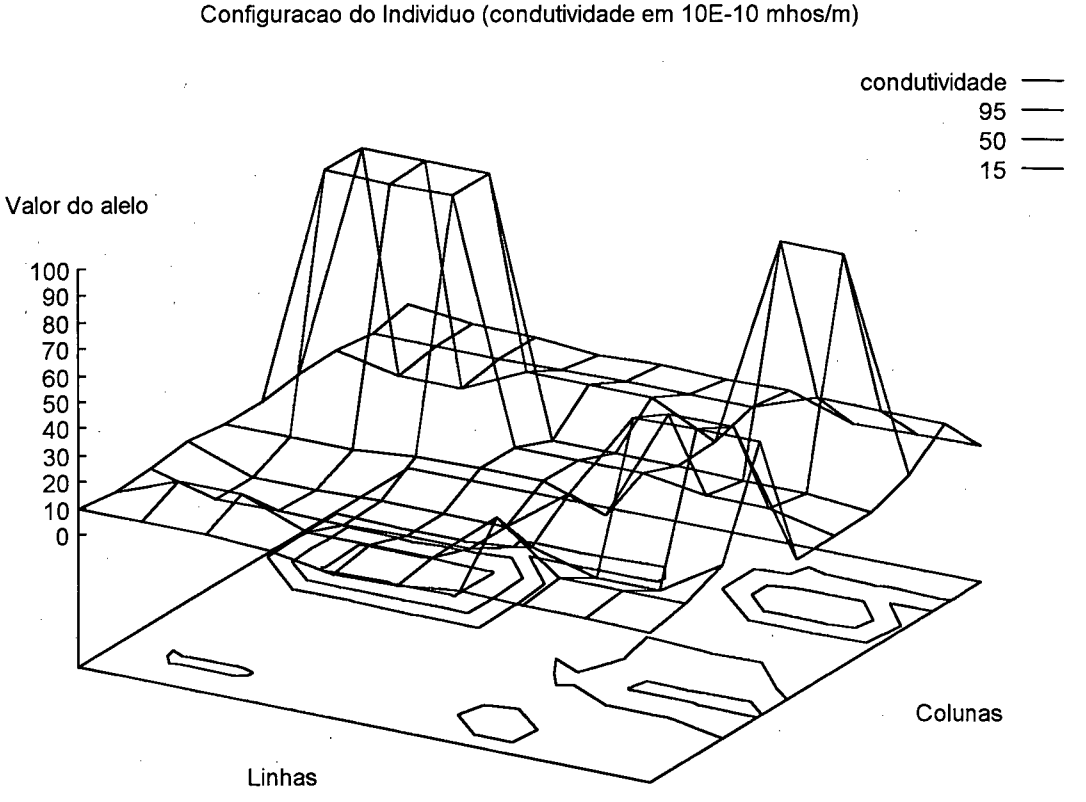


Fig. 24: Rodada clássica: ruído de 3%

Melhor indivíduo com erro gaussiano de 4%

8.7	7.4	9.0	9.6	7.5	8.4	11.0	12.0	8.4	12.9
15.4	16.0	2.0	2.8	8.4	100.0	100.0	5.0	9.2	13.0
3.4	5.6	1.3	.7	8.2	100.0	37.4	18.5	9.7	3.5
3.3	7.0	.9	2.7	84.6	3.0	3.8	24.0	12.4	4.3
10.9	25.8	6.6	24.8	2.5	54.3	14.2	4.7	9.1	13.3
1.9	10.3	.3	18.7	100.0	5.8	1.3	100.0	18.8	.4
7.5	8.3	15.1	100.0	.7	1.1	2.9	5.5	18.1	1.5

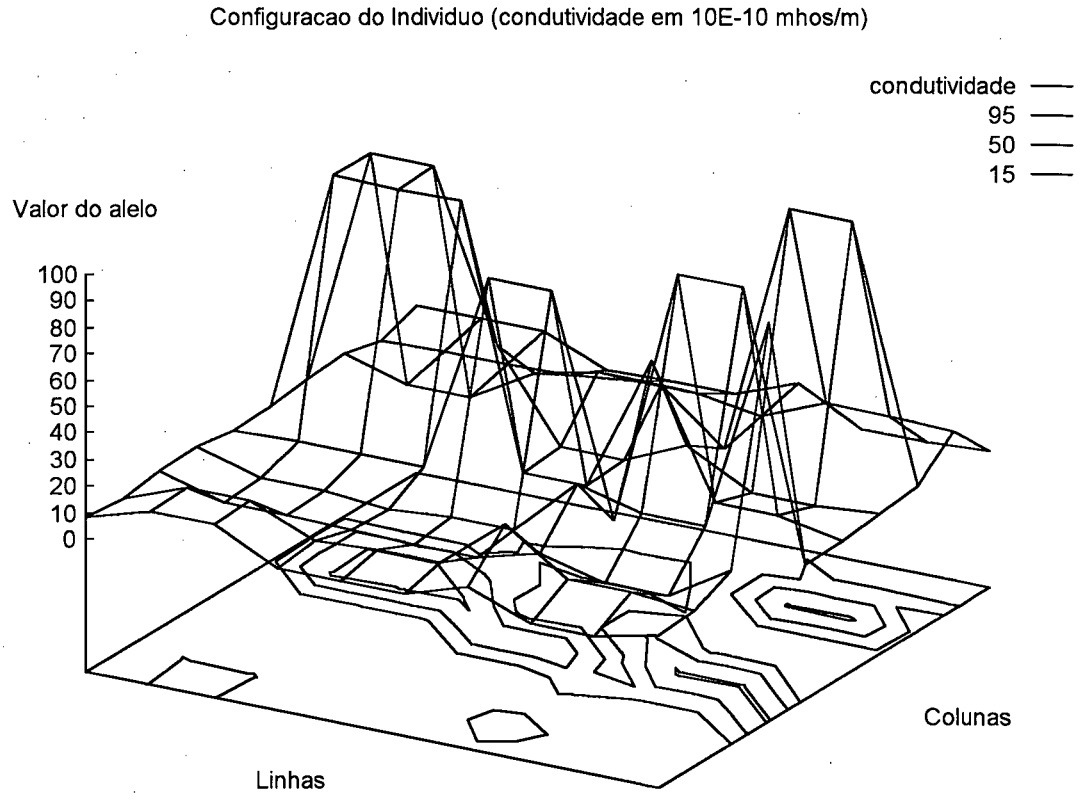


Fig. 25: Rodada clássica com 4% de ruído

Melhor indivíduo com erro gaussiano de 8%

7.3	6.4	10.8	7.3	5.9	4.3	20.3	10.2	9.3	13.6
20.2	19.1	1.6	9.1	8.0	57.6	100.0	9.2	5.9	16.4
.6	4.6	.1	.3	3.2	80.1	58.4	26.7	6.8	3.0
7.2	12.4	3.7	2.3	84.4	9.0	5.2	27.2	30.0	5.2
22.2	43.9	1.0	60.3	1.7	14.6	.1	11.8	1.7	53.4
.3	4.5	.8	60.2	100.0	4.2	7.7	54.4	100.0	.1
4.7	10.8	99.9	45.2	.6	.3	.3	6.5	18.7	1.0

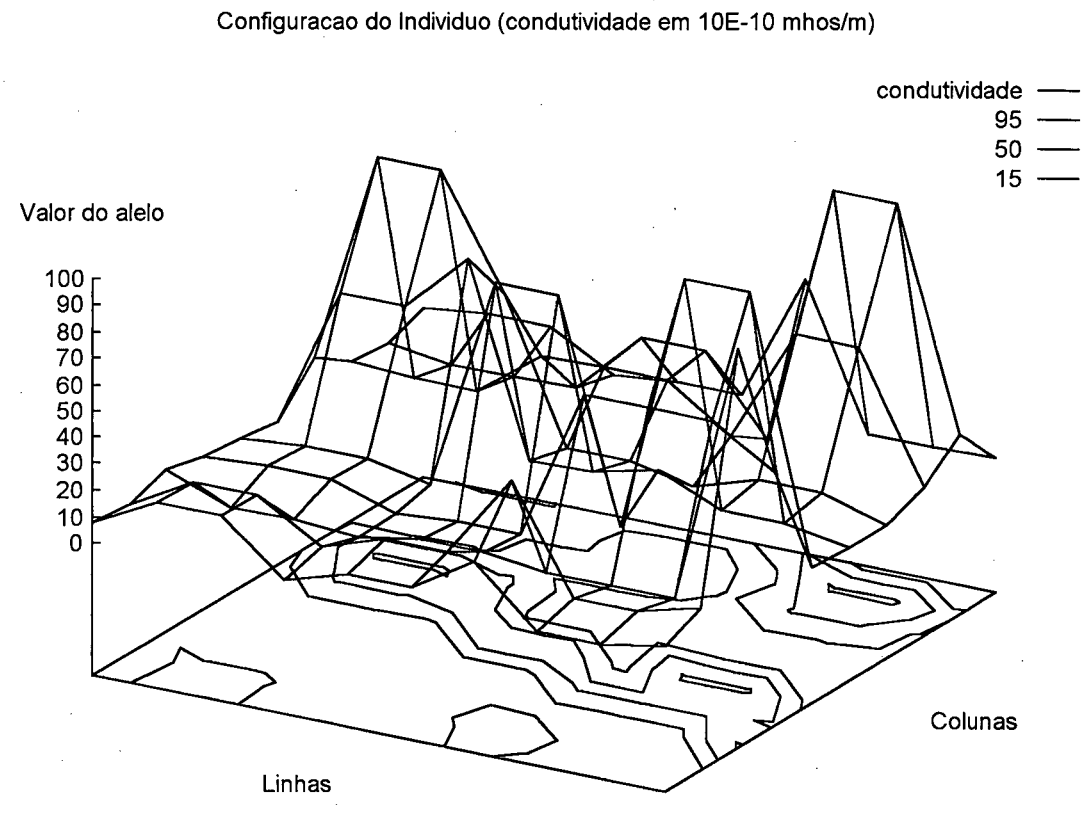


Fig. 26: Rodada clássica com ruído de 8%

Estes testes foram conduzidos com os valores de $\gamma_0 = 0.00$ e $\gamma_1 = 0.02$ que conforme se viu anteriormente constituem a dupla de valores com melhor desempenho para este problema. Percebe-se degradação crescente, na medida em que o ruído aumenta. Considerando-se o limite (arbitrário) que aqui se impôs, de que só existe convergência quando o erro condutivo é inferior a 1000×10^{-11} mhos/m, pode-se afirmar que no último caso não há convergência.

5.4.4 Níveis diferenciados de ruído - engenho evolutivo

Conduziram-se testes similares aos anteriores, para estabelecer a imunidade ao ruído, agora com o engenho evolutivo. Os testes usaram a configuração 1, com níveis diferenciados de ruído gaussiano, 3 medidas distintas, com o valor médio entre as três. O ciclo de otimização era chamado a cada 10 gerações e cada ciclo era composto por 5+geração / 10 etapas de busca local e 3 etapas de homogeneização. 100% dos indivíduos era inicializado com o valor 10. Usou-se elitismo e o operador de recombinação é o SPAUC em 100% dos casos com 2 cortes horizontais e 3 verticais (12 regiões), com probabilidade de homogeneização dos descendentes igual a 5%. Foram 150 gerações, com 100 indivíduos na população, probabilidades de crossover e mutação de 85% e 3% respectivamente, com tamanho de torneio de 15 indivíduos. A tabela 41 resume os parâmetros.

Tabela 41 - Parâmetros dos testes de imunidade ao ruído (engenho evolutivo)

Parâmetro	Valor
Configuração utilizada	1
Distribuição da população inicial	100% igual a 10
Ruído	1; A=3; T=médio; R=0.02
Otimização externa	1 a cada 10 gerações; local=5+gen/10;homo=3
Elitismo	1
Crossover	1.0 SPAUC; L=2; C=3; F1=0.05; F2=0.05
Parâmetros da rodada	G=150; P=100; PC=0.85; PM=0.03; T=15

Os resultados

Tabela 42 - Resultados do processamento de ruído

Rodada	Erro magnético	Erro condutivo
2 % ruído	0.39	197.1
2 %	0.61	501.9
2 %	0.64	359.8
Média	0.54	352.9
3 %	0.87	453.7
3 %	1.02	709.3
3 %	0.73	258.6
Média	0.87	473.8
4 %	0.87	301.6
4 %	0.92	360.3
4 %	1.42	747.1
Média	1.07	469.6
8 %	1.85	830.8
8 %	1.58	610.3
8 %	1.83	503.2
Média	1.75	648.1

Para comparação, eis os indivíduos

Melhor indivíduo obtido no engenho evolutivo para 2% de ruído

9.7	10.3	9.4	10.1	9.9	9.7	9.6	10.4	11.3	9.2
8.6	11.4	1.0	1.0	11.4	84.7	100.0	9.1	8.6	11.8
12.5	9.3	1.0	1.3	9.1	74.0	100.0	9.6	11.4	11.6
12.3	9.6	9.9	7.6	6.4	6.9	12.2	12.3	11.2	9.4
7.4	10.8	8.5	11.7	12.2	11.7	13.7	11.8	9.4	7.1
14.0	21.0	6.7	5.9	11.9	6.9	2.7	7.5	10.9	21.3
5.1	7.4	12.5	12.1	7.1	6.9	12.3	27.4	12.7	2.5

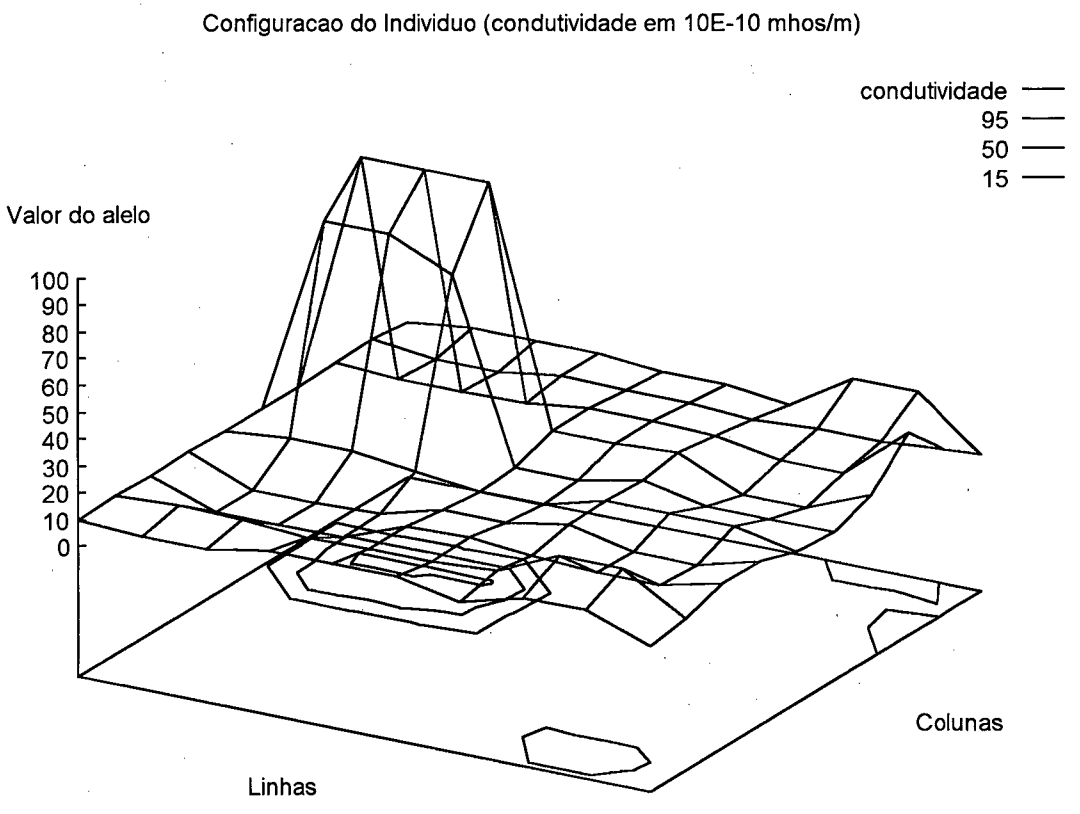


Fig. 27: Rodada evolutiva, melhor indivíduo com ruído de 2%

Melhor indivíduo obtido no engenho evolutivo para 3% de ruído

9.3	10.9	8.6	10.7	9.5	10.5	9.3	10.7	11.4	9.1
9.5	8.1	1.3	1.0	12.4	86.6	91.1	10.0	8.2	12.3
12.6	11.6	1.5	1.1	6.8	98.3	96.6	10.1	11.2	13.8
13.5	10.6	2.8	7.7	8.7	6.6	20.2	10.4	11.2	9.8
6.2	12.1	11.7	7.6	19.3	13.2	10.9	13.2	9.8	5.7
19.2	17.5	1.2	3.2	7.5	7.0	2.1	6.3	15.4	20.4
3.2	6.2	23.8	8.0	4.8	6.2	12.4	34.4	12.9	1.2

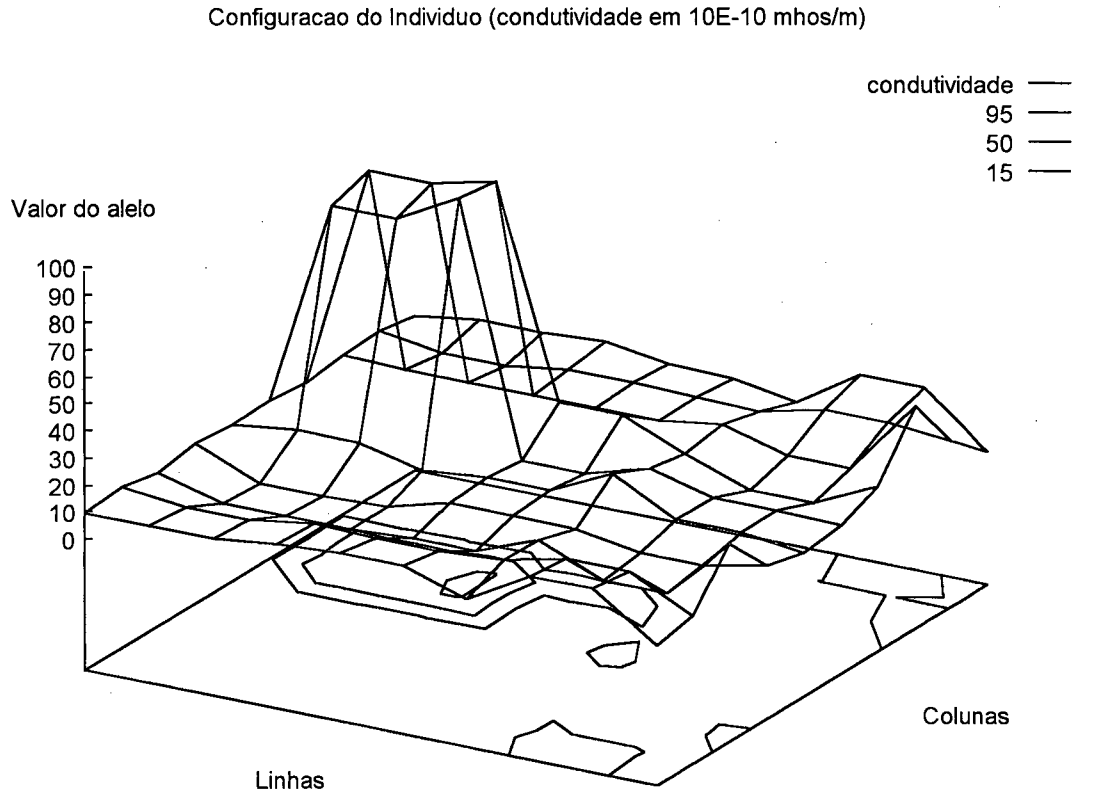


Fig. 28: Rodada evolutiva, melhor indivíduo com ruído de 3%

Melhor indivíduo obtido no engenho evolutivo para 4% de ruído

9.2	11.1	9.0	10.2	9.7	10.0	9.1	10.8	12.2	8.4
7.7	10.7	1.0	1.0	13.1	78.3	100.0	8.1	8.4	13.2
13.0	17.7	1.2	1.4	6.0	96.9	92.9	9.8	10.6	16.3
12.0	6.1	10.5	6.4	8.5	7.1	10.7	16.3	9.0	9.9
8.2	9.7	10.0	10.9	16.4	13.7	14.4	17.0	8.1	5.8
15.9	45.9	1.2	6.4	5.7	8.6	1.9	4.2	16.7	24.0
2.6	4.8	22.9	8.6	5.7	4.0	20.1	27.7	14.1	1.1

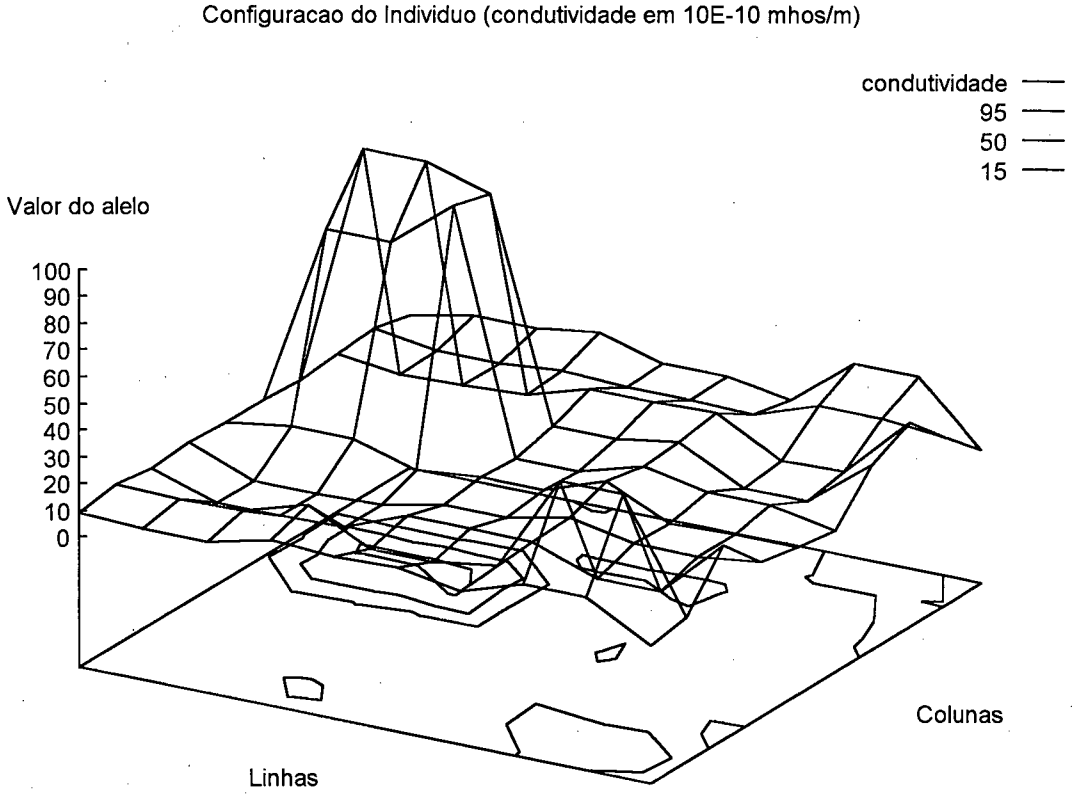


Fig. 29: Rodada evolutiva, melhor indivíduo com 4% de ruído

Melhor indivíduo obtido no engenho evolutivo para 8% de ruído

8.6	12.0	7.9	10.4	9.7	8.7	8.7	11.8	13.9	6.5
6.0	11.6	1.0	1.1	9.5	85.4	89.4	8.9	7.6	17.8
20.2	8.2	1.0	1.7	1.1	73.2	81.6	9.2	10.8	15.9
10.9	7.5	6.9	23.6	3.8	52.4	59.6	16.6	10.4	9.2
3.6	14.9	6.4	13.3	17.1	2.6	11.9	17.5	11.3	1.7
23.6	58.6	9.2	1.2	3.4	17.6	1.0	4.1	12.3	28.4
1.0	1.4	35.3	19.6	2.9	2.1	15.2	100.0	14.2	1.0

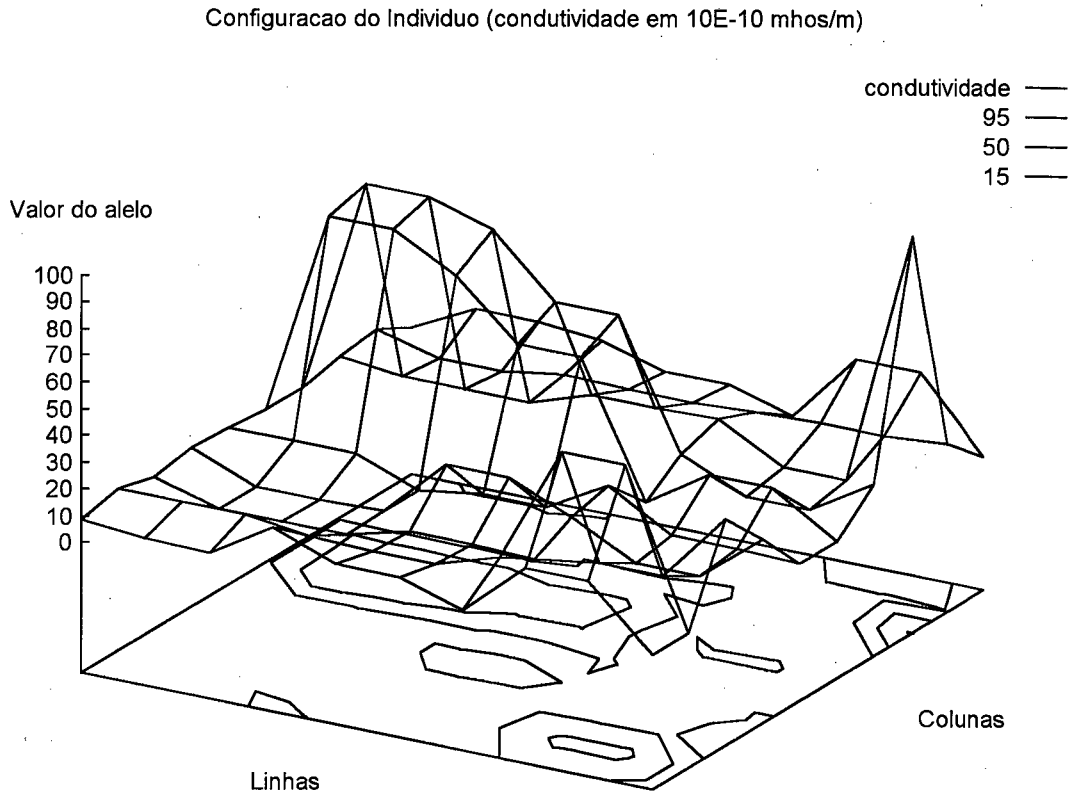


Fig. 30: Rodada evolutiva, melhor indivíduo com 8% de ruído

Repete-se aqui a conclusão obtida na comparação entre o processo clássico e o evolutivo no limiar de 1% de ruído. Novamente o resultado clássico é melhor individualmente falando e para baixos níveis de ruído. Entretanto à media em que o ruído aumenta, os resultados evolutivos começam a ser melhores. Por exemplo, em 4% a diferença é de 752.5 x 10^{-11} mhos/m (clássico) contra 469.6 x 10^{-11} mhos/m (evolutivo). Em 8%, os resultados são de 1051.3 x 10^{-11} mhos/m (clássico) versus 648.1 x 10^{-11} mhos/m. Note-se finalmente, que para o caso evolutivo é mister trabalhar com as médias de vári-

as rodadas, no caso 3. Se esta exigência for relaxada, e se se trabalhar com o melhor indivíduo, os resultados são ainda mais contundentes.

5.4.5 Caso 3 - otimização via gradiente mais regularização

Os testes agora referem-se a outras configurações.
Começa-se com a otimização clássica, para a configuração 3. Todos os valores iniciais são iguais a rocha, o ruído é de 1%. Os parâmetros estão na tabela 43.

Tabela 43 : Configuração 3 - otimização via gradiente mais regularização

Parâmetro	Valor
Configuração utilizada	3
Distribuição da população inicial	100% igual ao valor 10
Ruído	0.01

Os resultados

Tabela 44 : Resultados da configuração 3 (abordagem com otimização via gradiente mais regularização)

Rodada	Erro magnético	Erro condutivo
não conhecido	0.61	714.5
0.0 e 0.01	0.52	494.0
0.0 e 0.02	0.54	627.2
0.01 e 0.0	0.62	703.9
0.02 e 0.0	0.61	716.4

O caso 3, é a mais irregular das configurações. Neste caso o bloco de material de maior condutividade ocupa parte de todas as linhas, e este fato é notado nos resultados que são menos exatos do que aqueles obtidos em configurações mais regulares. O melhor resultado apresenta um erro condutivo de 494.0 x 10^{-11} mhos/m, e agora as configurações de γ_0 e γ_1 são ligeiramente diferentes.

5.4.6 Caso 3 - engenho evolutivo

Agora vem a comparação do caso com otimização mais regularização com o engenho evolutivo. Os testes usaram a configuração 3, com 1% de ruído gaussiano, 3 medidas distintas, com o valor médio entre as três. O ciclo de otimização era chamado a cada 10 gerações e cada ciclo era composto por 5+geração / 10 etapas de busca local e 3 etapas de homogeneização. 100% dos indivíduos era inicializado com o valor 10. Usou-se elitismo e o operador de recombinação é o SPAUC em 100% dos casos com 2

cortes horizontais e 3 verticais (12 regiões), com probabilidade de homogeneização dos descendentes igual a 5%. Foram 150 gerações, com 100 indivíduos na população, probabilidades de crossover e mutação de 85% e 3% respectivamente, com tamanho de torneio de 15 indivíduos. A tabela 45 resume os parâmetros.

Tabela 45 : Configuração 3 - engenho evolutivo

Parâmetro	Valor
Configuração utilizada	3
Distribuição da população inicial	100% igual a 10
Ruído	1; A=3; T=médio; R=0.01
Otimização externa	1 a cada 10 gerações; local=5+gen/10;homo=3
Elitismo	1
Crossover	1.0 SPAUC; L=2; C=3; F1=0.05; F2=0.05
Parâmetros da rodada	G=150; P=100; PC=0.85; PM=0.03; T=15; SA=0.111

Tabela 46 : Resultados da configuração 3

Rodada	Erro magnético	Erro condutivo
1	0.17	537.1
2	0.19	185.5
3	0.31	619.5
4	0.24	501.9
5	0.23	591.1
6	0.24	418.1
7	0.23	571.8
Média	0.23	489.2

Comparando-se o melhor resultado da abordagem clássica para esta configuração, que é de 494.0 x 10^{-11} mhos/m, com a média das diversas execuções do engenho evolutivo que é de 489.2 x 10^{-11} mhos/m, verifica-se pequena vantagem para o segundo. Novamente, como antes, se a comparação for feita com o melhor resultado (185.5 x 10^{-11} mhos/m) a vantagem tornar-se-á significativamente maior.

5.4.7 Caso 4 - otimização via gradiente mais regularização

Os testes agora referem-se a configuração 4, que é mais regular de todas. Dos 70 blocos que compõe a solução buscada, 69 deles têm condutividade igual a da rocha e apenas um único prisma tem condutividade dez vezes menor.

Começa-se com a otimização clássica, para a configuração 4. Todos os valores iniciais são iguais a rocha, o ruído é de 1%. Os parâmetros estão na tabela 47.

Tabela 47 : Configuração 4 - otimização via gradiente mais regularização	
Parâmetro	Valor
Configuração utilizada	4
Distribuição da população inicial	100% igual ao valor 10 no primeiro caso e um valor aleatório (escolhido dentre 16 possíveis) no segundo caso
Ruído	0.01

Os resultados

Tabela 48 : Resultados da configuração 4 (abordagem clássica) - inicialização com condutividade da rocha		
Rodada	Erro magnético	Erro condutivo
não conhecido	10.8	247.6
não conhecido	0.37	32.3
não conhecido	0.71	48.5
0.0 e 0.01	0.46	49.3
0.0 e 0.02	10.5	92.0
0.01 e 0.0	10.6	135.4
0.02 e 0.0	13.4	849.0
Inicialização com valores randômicos		
0.0 e 0.03	36.3	1223.6
0.03 e 0.0	34.6	1332.2

O único fato digno de registro aqui é que quando não há inicialização com o valor da rocha, não há convergência, o que é representado por erros condutivos superiores a 1000×10^{-11} mhos/m.

5.4.8 Caso 4 - engenho evolutivo

Agora vem a comparação com o engenho evolutivo. Os testes usaram a configuração 4, com 1% de ruído gaussiano, 3 medidas distintas, com o valor médio entre as três. O ciclo de otimização era chamado a cada 10 gerações e cada ciclo era composto por 5+geração / 10 etapas de busca local e 3 etapas de homogeneização. 100% dos indivíduos era inicializado com o valor 10. Usou-se elitismo e o operador de recombinação é o SPAUC em 100% dos casos com 2 cortes horizontais e 3 verticais (12 regiões), com probabilidade de homogeneização dos descendentes igual a 5%. Foram 150 gerações, com 100 indivíduos na população, probabilidades de crossover e mutação de 85% e 3% respectivamente, com tamanho de torneio de 15 indivíduos. A tabela 49 resume os parâmetros.

Tabela 49 : Configuração 4 - engenho evolutivo	
Parâmetro	Valor
Configuração utilizada	4

Distribuição da população inicial	100% igual a 10
Ruído	1; A=3; T=médio; R=0.01
Otimização externa	1 a cada 10 gerações; local=5+gen/10;homo=3
Elitismo	1
Crossover	1.0 SPAUC; L=2; C=3; F1=0.05; F2=0.05
Parâmetros da rodada	G=150; P=100; PC=0.85; PM=0.03; T=15

Tabela 50 : Resultados da configuração 4 - inicialização com a condutividade da rocha

Rodada	Erro magnético	Erro condutivo
1	0.15	61.1
2	0.17	68.0
3	0.55	150.4
4	0.21	56.4
5	0.38	136.7
6	0.28	82.2
Média	0.29	92.4
Inicialização randômica - os 70 valores são randômicos		
1	0.13	49.9
2	0.19	56.5

Em valores isolados, o resultado clássico é melhor (32.3 x 10^{-11} mhos/m versus 92.4 x 10^{-11} mhos/m -- na média). Entretanto, quando a inicialização não se dá, ou por outro lado, quando não se puder fazer inferência sobre os valores de condutividade no entorno, a solução evolutiva prova seu valor. Pode-se sugerir que neste caso o resultado é fruto da atuação do operador de regularização, que rapidamente "espalha" os valores corretos para o entorno, ainda que eles não fossem de antemão conhecidos.

5.4.9 Caso 5 - aleatórios

O caso 5 não foi estudado na abordagem original (Ramos e Campos Velho, 1996) e foi incluída aqui apenas com a finalidade de estudar o desempenho do algoritmo evolutivo.

Rodaram-se 2 execuções e em ambas não houve convergência. Note-se que embora se fale em inicialização aleatória apenas 16 valores aleatórios eram usados (por restrição do modelo originalmente utilizado).

Já na abordagem evolutiva também houve 2 execuções usando as mesmas condições das demais, a saber: 1% de ruído gaussiano, 3 medidas distintas, com o valor médio entre as três. O ciclo de otimização era chamado a cada 10 gerações e cada ciclo era composto por 5+geração / 10 etapas de busca local e 3 etapas de homogeneização. 100% dos indivíduos era inicializado com o valor 10. Usou-se elitismo e o operador de

recombinação é o SPAUC em 100% dos casos com 2 cortes horizontais e 3 verticais (12 regiões), com probabilidade de homogeneização dos descendentes igual a 5%. Foram 150 gerações, com 100 indivíduos na população, probabilidades de crossover e mutação de 85% e 3% respectivamente, com tamanho de torneio de 15 indivíduos.

O resultado alcançado foi:

Tabela 51 : Resultados da configuração 5 (engenho evolutivo)

Rodada	Erro magnético	Erro condutivo
não conhecido	0.41	1244.7
não conhecido	0.41	1652.8
Média	0.41	1448.7

Conclusão

A basear-se no critério que informa que erros condutivos superiores a 1000 (x 10^{-11} mhos/m) caracterizam não convergência, forçoso é se constatar que não houve convergência neste caso.

5.5 Conclusão sobre os resultados

Da análise dos resultados numéricos aqui apresentados, podem-se extrair as seguintes conclusões:

a) Para situações onde o corpo de material diferente inserido na rocha é regular, e com baixo nível de ruído, o engenho clássico é melhor

b) Em situações onde o ruído cresce, o engenho evolutivo é melhor

c) Quando o corpo inserido é irregular, o engenho evolutivo é melhor.

d) Finalmente, pode-se afirmar que o engenho evolutivo é mais robusto.

e) Em contrapartida, o engenho evolutivo é sempre mais consumidor de recursos de máquina, demandando em média de 2 a 5 vezes mais chamadas à função objetivo

5.6 Melhoramentos

Neste tópico descrevem-se melhoramentos que foram sendo introduzidos no processo de busca à medida em que este avançava. Eles estão separados do texto para não "contaminar" os testes que lá foram efetuados e permitir a comparação dos diversos resultados encontrados.

5.6.1 Busca local V2

Uma segunda implementação de busca local (conhecida no escopo do trabalho como "busca local V2") também foi implementada. O motivo que levou a esta segunda versão foi a constatação de que em certas ocasiões – tipicamente ao final dos ciclos evolutivos, a busca local V1 levava a uma perda de valor na função objetivo. Nesse sentido, deixava de ser uma subida da encosta para se transformar na descida da encosta. A explicação para o fato residia em que, embora os saltos considerados individualmente levassem a uma melhora na função objetivo, quando aplicados todos ao mesmo tempo, em conjunto, levavam a uma piora deste valor. Este fenômeno é conhecido no âmbito de computação evolutiva como *epistasia*, o que pode ser traduzido pela influência que certos alelos exercem sobre seus vizinhos.

A versão 2, corrigiu esta anomalia, aplicando o salto, a cada vez que ele era julgado e verificava-se que ele conduzia a uma melhora na função objetivo. Acompanhe a aplicação no exemplo:

Seja um cromossomo qualquer

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24

que como se sabe tem um fitness F_1.

Após a geração de um infinitesimal, ele é somado à primeira célula, como em

1+δ	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24

Após esta soma, a função objetivo é recalculada, tendo como resultado F_1'. Se F_1' é maior que F_1, δ é imediatamente acrescentado ao indivíduo, antes de fazer-se o teste do próximo valor. Caso não tenha havido melhoria na função objetivo, repete-se o teste agora usando o mesmo infinitésimo com valor negativo, por exemplo

1-δ	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24

E assim prossegue a busca local V2 até o final do indivíduo. Esta aplicação (V2) também aumenta o valor do infinitésimo aplicado à medida em que a profundidade aumenta multiplicando-o pela profundidade do valor que está a ser testado.

Não houve diferenças quantitativas importantes entre as buscas locais V1 e V2. Um teste inicial entre ambas pode ser mostrado a seguir. A primeira execução repete uma anterior, cujo melhor resultado foi de

```
 10.1   9.8   9.6  10.1   9.7  10.0   9.7  10.2  10.6   9.6
  8.5  12.0   1.2   1.1  10.5  99.0  97.1   9.8   9.1  11.2
 11.7   7.6   1.1   1.0   7.2  99.9  99.8  10.2  10.3  11.0
 14.5   9.2   7.3   9.9  18.8   6.8  11.0  11.5  10.8   9.6
  8.5  10.6  12.7   8.3  16.2  11.0  12.3  10.6   9.5   8.7
  9.7  20.1   3.6   8.7  12.1  13.6   5.1   9.5  11.6  13.9
 14.4   3.0  46.4   5.3   5.8   9.2  15.7  15.2   8.8   6.2
```

e cujos resultados foram de Em=0.29 e Ec=172.2. O desempenho desta rodada (a 204) pode ser vista no gráfico

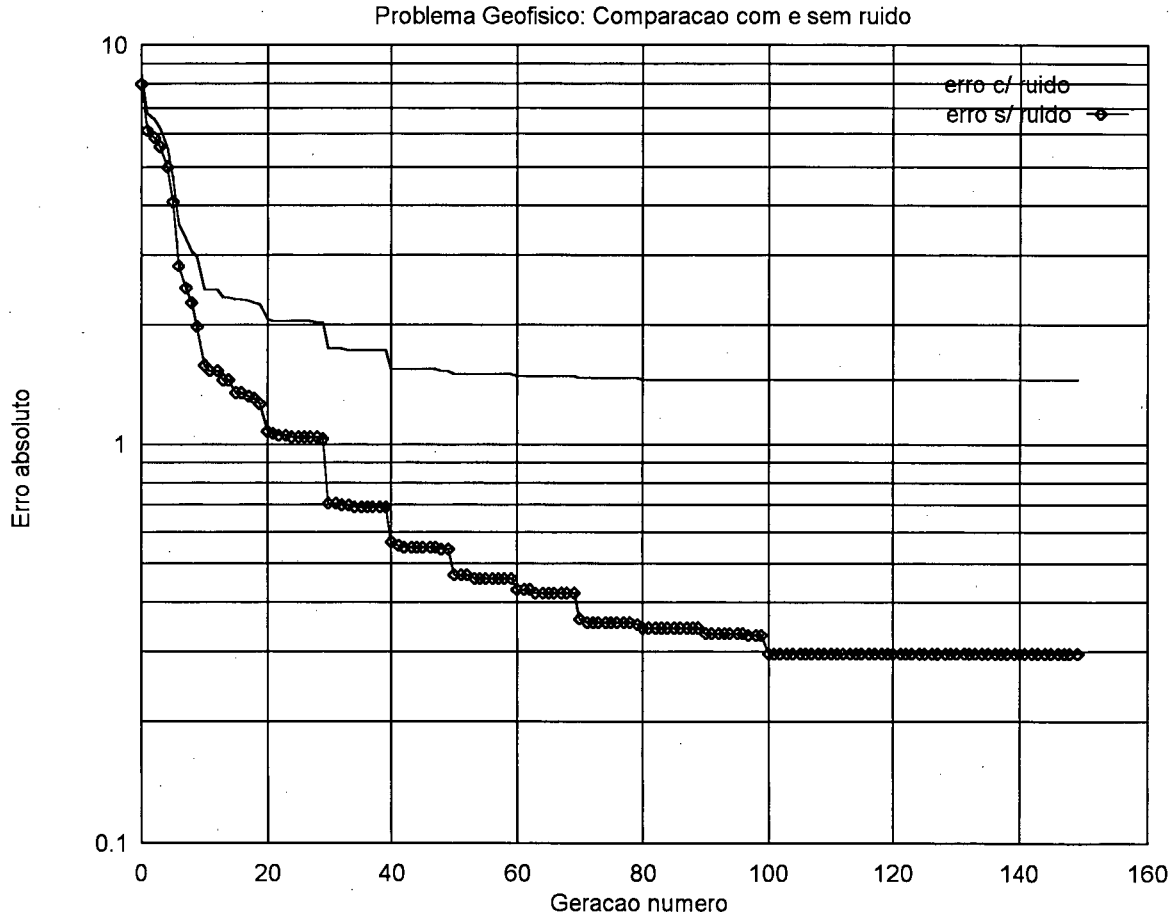


Fig. 31: Erro magnético em execução convencional (V1)

Na seqüência, o algoritmo básico foi modificado, chamando-se um ciclo de busca local a cada 2 gerações, com um ciclo de 3 etapas de busca local, sem homogeneização e sem SPAUC (ou seja com crossover uniforme)

Os resultados foram: Em=0.73, Ec = 414.0 e o melhor indivíduo usando a busca local versão 1.

11.0	12.9	6.5	7.2	10.1	7.4	11.0	10.0	10.8	9.4
6.7	1.0	10.8	4.8	4.3	100.0	99.0	12.3	7.5	13.3
14.4	18.4	1.0	1.0	9.3	63.7	94.4	10.1	9.0	11.7
8.3	22.9	1.0	1.0	48.7	5.3	7.5	11.6	16.5	10.1
8.0	7.1	13.5	9.3	8.3	3.4	38.8	4.6	22.9	3.6
13.7	20.2	3.9	10.5	4.1	20.9	1.5	14.6	12.2	13.2
6.7	7.9	8.3	28.2	4.6	4.1	20.3	41.5	7.2	4.8

O resultado desta rodada (303) pode ser visto no gráfico

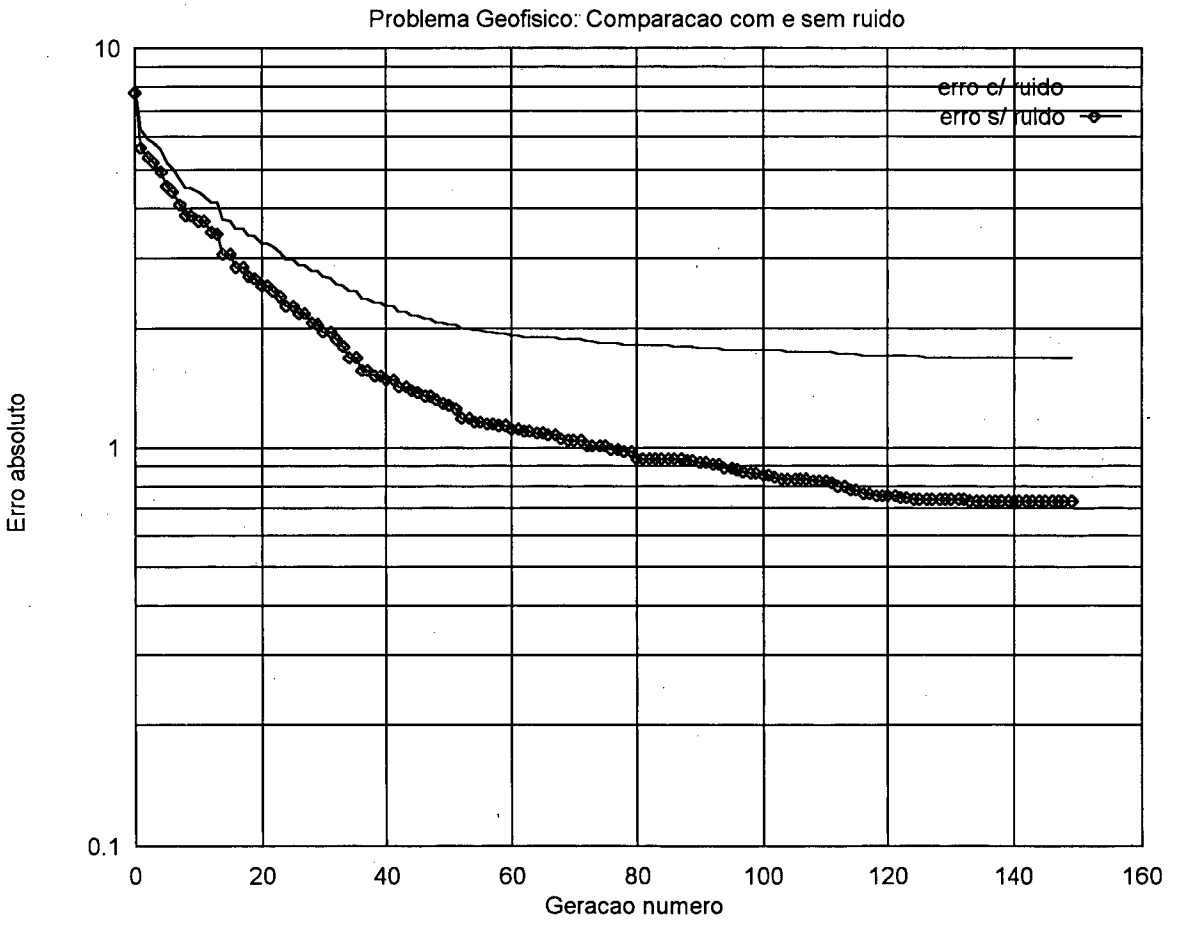


Fig. 32: Erro magnético em rodada especial usando busca local V1

Os resultados foram Em = 0.72 e Ec= 443.0 e o melhor indivíduo usando a busca local versão 2.

11.1	12.8	6.2	7.1	10.2	7.5	11.2	9.8	11.1	9.2
6.0	1.0	13.6	4.5	4.4	100.0	97.2	13.6	6.1	15.1
15.8	20.2	1.0	1.0	9.6	67.1	70.5	11.7	10.5	11.7
6.8	19.5	1.0	1.0	52.6	5.6	1.6	8.9	14.8	8.9
8.4	6.7	15.5	9.6	7.8	1.5	44.9	6.1	17.5	4.7
12.7	24.8	1.8	13.8	2.6	20.3	1.0	12.2	12.7	12.5
5.1	9.8	15.0	11.8	3.2	9.7	9.7	47.1	7.2	5.4

O gráfico desta execução (0304) é:

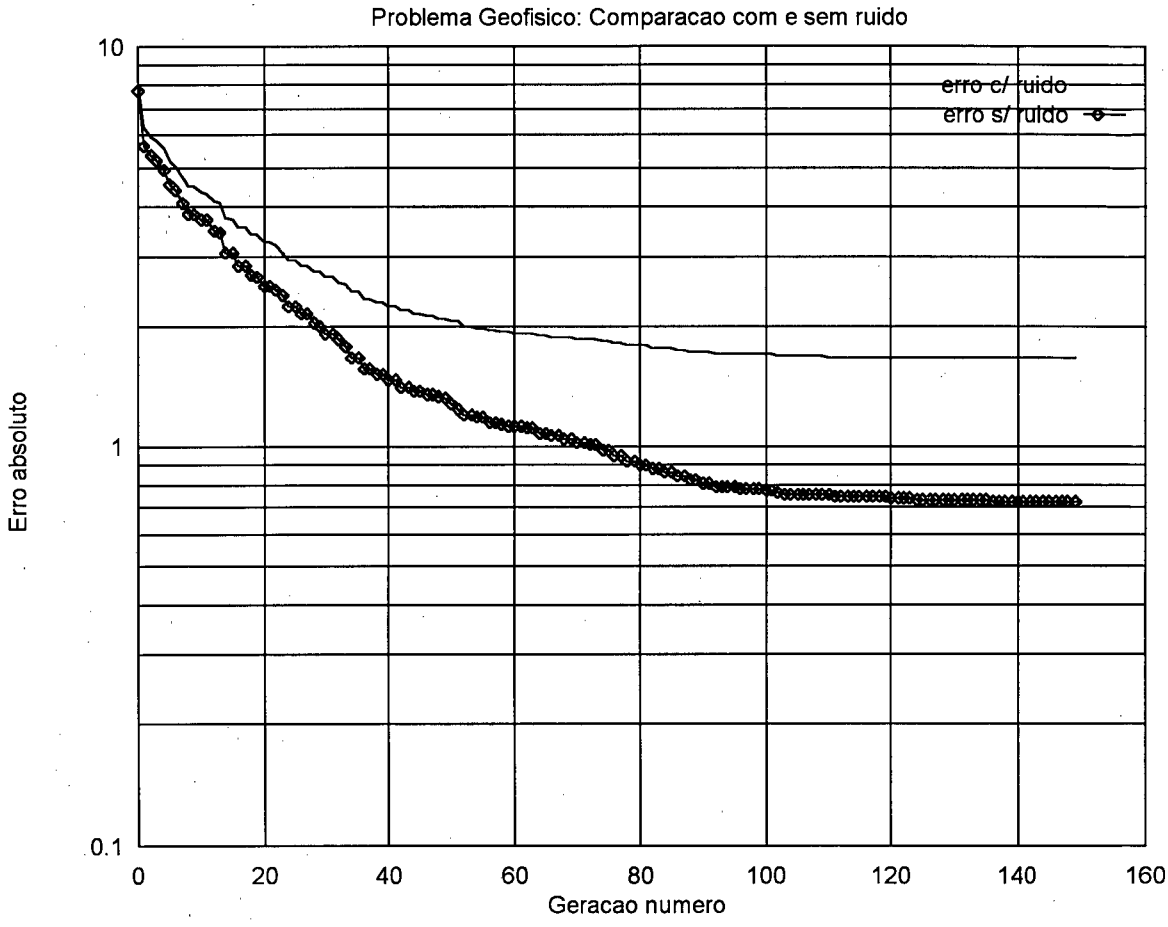


Fig. 33: Erro magnético em rodada especial usando busca local V2

Para efeito de comparação entre as versões 1 e 2 da busca local, segue-se um resultado numérico, obtido geração após geração, em 150 delas, para o mesmo problema. A primeira coluna representa a geração, a segunda o valor original da função objetivo (antes da busca local), a terceira coluna mostra quanto se avançou na busca local e finalmente a última coluna acumula a terceira, indicando qual foi o ganho acumulado geração a geração.

Tabela 52: Ganhos na busca local (versão 1)

ger.	valor da f.o.	ganho na b.l.	ganho acumulado

1	0.5582361817	0.0081943870	0.0081943870
3	0.5778698921	0.0054728985	0.0136672854
5	0.6196237206	0.0099763274	0.0236436129
7	0.6537989974	0.0168068409	0.0404504538
9	0.6600176096	0.0057278275	0.0461782813
11	0.6699826717	0.0059679151	0.0521461964
13	0.6870915890	0.0096437335	0.0617899299
15	0.7139490247	0.0116225481	0.0734124780
17	0.7229580283	0.0090090036	0.0824214816
19	0.7330567241	0.0100986958	0.0925201774
21	0.7373043895	0.0042476654	0.0967678428
23	0.7548869252	0.0095079541	0.1062757969
25	0.7615001798	0.0066132545	0.1128890514
27	0.7683685422	0.0068683624	0.1197574139
29	0.7749089599	0.0065404177	0.1262978315
31	0.7815377116	0.0064880252	0.1327858567
33	0.7912914753	0.0072950721	0.1400809288
35	0.7971391082	0.0058476329	0.1459285617
37	0.8030142188	0.0039206743	0.1498492360
39	0.8056847453	0.0026705265	0.1525197625
41	0.8104792237	0.0047944784	0.1573142409
43	0.8144294620	0.0039502382	0.1612644792
45	0.8184981346	0.0040686727	0.1653331518
47	0.8213495612	0.0028514266	0.1681845784
49	0.8229961395	0.0016465783	0.1698311567
51	0.8268578649	0.0033972263	0.1732283831
53	0.8290171623	0.0015639663	0.1747923493
55	0.8300769925	0.0010598302	0.1758521795
57	0.8313668370	0.0012898445	0.1771420240
59	0.8327359557	0.0013691187	0.1785111427
61	0.8343376517	0.0016016960	0.1801128387
63	0.8350694776	0.0007318258	0.1808446646
65	0.8357165456	0.0006470680	0.1814917326
67	0.8365159631	0.0007994175	0.1822911501
69	0.8372802138	0.0007642508	0.1830554008
71	0.8387148976	0.0014346838	0.1844900846
73	0.8392619491	0.0005470514	0.1850371361
75	0.8401403427	0.0008783937	0.1859155297
77	0.8405942917	0.0004539490	0.1863694787
79	0.8406818509	0.0000875592	0.1864570379
81	0.8414132595	0.0006017089	0.1870587468
83	0.8412182331	-0.0001950264	0.1868637204
85	0.8430161476	0.0010371804	0.1879009008
87	0.8433787823	0.0003626347	0.1882635355
89	0.8438973427	0.0005185604	0.1887820959
91	0.8445178866	0.0006205440	0.1894026399
93	0.8451007009	0.0005828142	0.1899854541
95	0.8456595540	0.0005588531	0.1905443072
97	0.8463187814	0.0006592274	0.1912035346
99	0.8463125229	-0.0000062585	0.1911972761
101	0.8466237187	0.0003111959	0.1915084720
103	0.8467343450	0.0001106262	0.1916190982
105	0.8473607898	0.0005160570	0.1921351552
107	0.8472270966	-0.0001336932	0.1920014620
109	0.8477199674	0.0003245473	0.1923260093
111	0.8486523628	0.0009018779	0.1932278872
113	0.8494486809	0.0007963181	0.1940242052
115	0.8499243855	0.0004757047	0.1944999099
117	0.8494615555	-0.0004628301	0.1940370798
119	0.8506687284	0.0004556775	0.1944927573
121	0.8505743146	-0.0000944138	0.1943983436
123	0.8511362672	0.0005619526	0.1949602962
125	0.8510354161	-0.0001008511	0.1948594451
127	0.8504543304	-0.0006405711	0.1942188740
129	0.8511373401	-0.0000451207	0.1941737533
131	0.8511325717	-0.0002146363	0.1939591169

133	0.8515803814	0.0001805425	0.1941396594
135	0.8516960740	0.0000746250	0.1942142844
137	0.8510956764	-0.0006098747	0.1936044097
139	0.8502110839	-0.0014849901	0.1921194196
141	0.8514401913	-0.0002451539	0.1918742657
143	0.8509311080	-0.0008023381	0.1910719275
145	0.8520801663	0.0003067255	0.1913786530
147	0.8519287705	-0.0001513958	0.1912272573
149	0.8514001966	-0.0006751418	0.1905521154

Para efeito de comparação, segue a mesma tabela para a busca local versão 2

Tabela 53 : Desempenho da busca local versão 2

ger.	valor da f.o.	ganho da b.l.	ganho acumul.
1	0.5582298636	0.0081880689	0.0081880689
3	0.5778517127	0.0054897070	0.0136777759
5	0.6197204590	0.0100089312	0.0236867070
7	0.6539440155	0.0169537663	0.0406404734
9	0.6602213979	0.0057814121	0.0464218855
11	0.6701598763	0.0060094595	0.0524313450
13	0.6874173284	0.0097869635	0.0622183084
15	0.7144234180	0.0118013024	0.0740196109
17	0.7234730721	0.0090496540	0.0830692649
19	0.7337879539	0.0103148818	0.0933841467
21	0.7381007075	0.0043127537	0.0976969004
23	0.7559909225	0.0093869567	0.1070838571
25	0.7626457810	0.0066548586	0.1137387156
27	0.7695377469	0.0068919659	0.1206306815
29	0.7760547996	0.0065170527	0.1271477342
31	0.7826445699	0.0064340830	0.1335818172
33	0.7924823761	0.0072504878	0.1408323050
35	0.7982165813	0.0057342052	0.1465665102
37	0.8043326735	0.0042418838	0.1508083940
39	0.8071663976	0.0028337240	0.1536421180
41	0.8115614057	0.0043950081	0.1580371261
43	0.8155911565	0.0039753914	0.1620125175
45	0.8175302148	0.0019390583	0.1639515758
47	0.8191887736	0.0016585588	0.1656101346
49	0.8221499920	0.0029612184	0.1685713530
51	0.8261975646	0.0038840175	0.1724553704
53	0.8288536668	0.0026561022	0.1751114726
55	0.8307716846	0.0019180179	0.1770294905
57	0.8314159513	0.0006442666	0.1776737571
59	0.8325245976	0.0011086464	0.1787824035
61	0.8335525990	0.0010280013	0.1798104048
63	0.8350692391	0.0014996529	0.1813100576
65	0.8360531926	0.0009839535	0.1822940111
67	0.8369665146	0.0009133220	0.1832073331
69	0.8376653790	0.0006988645	0.1839061975
71	0.8387406468	0.0010752678	0.1849814653
73	0.8400311470	0.0012905002	0.1862719655
75	0.8414857388	0.0014545918	0.1877265573
77	0.8427577019	0.0012719631	0.1889985204
79	0.8436866403	0.0009289384	0.1899274588
81	0.8451445699	0.0014579296	0.1913853884
83	0.8460513949	0.0009068251	0.1922922134
85	0.8471778035	0.0011264086	0.1934186220
87	0.8483341336	0.0011563301	0.1945749521
89	0.8490262628	0.0006921291	0.1952670813
91	0.8496952057	0.0006689429	0.1959360242
93	0.8503582478	0.0004308820	0.1963669062
95	0.8505741954	0.0002159476	0.1965828538
97	0.8508736491	0.0002994537	0.1968823075
99	0.8512020111	0.0003283620	0.1972106695
101	0.8515393138	0.0003373027	0.1975479722

103	0.8519496918	0.0004103780	0.1979583502
105	0.8521208763	0.0001645088	0.1981228590
107	0.8522667885	0.0001459122	0.1982687712
109	0.8524526954	0.0001859069	0.1984546781
111	0.8525743484	0.0001216531	0.1985763311
113	0.8526547551	0.0000804067	0.1986567378
115	0.8527842164	0.0001294613	0.1987861991
117	0.8529863358	0.0002021194	0.1989883184
119	0.8531398773	0.0001535416	0.1991418600
121	0.8532500267	0.0001101494	0.1992520094
123	0.8533666730	0.0001166463	0.1993686557
125	0.8534286022	0.0000619292	0.1994305849
127	0.8534509540	0.0000223517	0.1994529366
129	0.8534620404	0.0000110865	0.1994640231
131	0.8534784913	0.0000164509	0.1994804740
133	0.8535301089	0.0000516176	0.1995320916
135	0.8535978198	0.0000677109	0.1995998025
137	0.8536525369	0.0000547171	0.1996545196
139	0.8537244797	0.0000719428	0.1997264624
141	0.8537711501	0.0000466704	0.1997731328
143	0.8538214564	0.0000503063	0.1998234391
145	0.8538391590	0.0000177026	0.1998411417
147	0.8538877368	0.0000485778	0.1998897195
149	0.8539111614	0.0000234246	0.1999131441

Conclusão: percebeu-se uma melhora pequena quando V2 é aplicada em comparação a V1. Esta melhora é no presente caso de menos de 0,2% sobre o valor da função objetivo. Quando considerado o ganho introduzido pela busca local, percebe-se um ganho de 1% quando usou-se a V2.

5.6.2 Mutação dependente da profundidade

Para este teste, modificou-se a função de mutação de campo, passando a utilizar-se a função de mutação, ligeiramente modificada para que, quanto maior a profundidade, maior fosse o salto aleatório gerado pela mutação. A justificativa para essa estratégia é que, estudos prévios do modelo em estudo, garantem que, quanto menor a profundidade, maior é a influência de eventuais mutações sobre o valor da aptidão. Ao se aplicar o mesmo valor de mutação, à medida em que a profundidade cresce, este valor único de mutação passa a representar melhoras (ou pioras) cada vez menores. Para a correção desta anomalia, criou-se esta mutação V2 que aumenta à medida em que a profundidade cresce.

O código que implementou este conceito foi:

```
i = 0
enquanto i < 70
        se (i < 10)
                txmutaprof = pmutation / 1.0
        fim-se
        se (i ≥ 10) e (i < 20)
```

```
                txmutaprof = pmutation / 1.0
        fim-se
        se (i ≥ 20) e (i < 30)
                txmutaprof = pmutation / 0.9
        fim-se
        se (i ≥ 30) e (i < 40)
                txmutaprof = pmutation / 0.9
        fim-se
        se (i ≥ 40) e (i < 50)
                txmutaprof = pmutation / 0.8
        fim-se
        se (i ≥ 50) e (i < 60)
                txmutaprof = pmutation / 0.8
        fim-se
        se (i ≥ 60) e (i < 70)
                txmutaprof = pmutation / 0.8
        fim-se
        i = i + 1
fim-enquanto
```

e também

```
i = 0
enquanto i < 70
        se (i < 10)
                txmutaprof = pmutation / 3.0
        fim-se
        se (i ≥ 10) e (i < 20)
                txmutaprof = pmutation / 2.6
        fim-se
        se (i ≥ 20) e (i < 30)
                txmutaprof = pmutation / 2.2
        fim-se
        se (i ≥ 30) e (i < 40)
                txmutaprof = pmutation / 1.8
        fim-se
        se (i ≥ 40) e (i < 50)
                txmutaprof = pmutation / 1.4
        fim-se
        se (i ≥ 50) e (i < 60)
                txmutaprof = pmutation / 1.0
        fim-se
        se (i ≥ 60) e (i < 70)
                txmutaprof = pmutation / 1.0
        fim-se
        i = i + 1
fim-enquanto
```

Todos os testes até aqui conduzidos tiveram uma taxa de mutação única de 3% independente da profundidade. Para os testes desta nova mutação (a V2), usaram-se duas configurações: A primeira (chamada A) implementa 3% para as camadas superiores e aumenta este valor em direção aos valores mais profundos de cada indivíduo. Já a configuração B, mantém os 3% para os valores mais profundos do indivíduo, diminuindo-se a taxa para os valores mais próximos à superfície.

Note-se que no primeiro caso (CONFIGURAÇÃO A), o piso é a taxa de mutação padrão utilizada (3%) que vale para as primeiras duas linhas. As duas seguintes utilizam uma taxa de 3.33 % e as três últimas, a taxa de 3.75%.

Já no segundo caso (CONFIGURAÇÃO B) o piso continua sendo 3%, mas aplicado às duas últimas linhas. A primeira linha tem taxa de 1%, a segunda de 1.15%, a terceira de

1.36%, a quarta de 1.66%, a quinta de 2.14%, e as sexta e sétima linhas têm taxa de 3%.

Em outras palavras, em (A) o valor anteriormente usado é o da primeira linha aumentando-se este valor para baixo. Em (B), o valor anteriormente usado é aplicado na última linha, diminuindo-se este valor à medida em que se sobe.

Os resultados alcançados foram:

Tabela 54 : Parâmetros dos testes de mutação dependente da profundidade

Parâmetro	Valor
Configuração utilizada	1
Distribuição da população inicial	100% igual a 10
Ruído	1; A=3; T=médio; R=0.01
Otimização externa	1 a cada 10 gerações; local=5+gen/10;homo=3
Elitismo	1
Crossover	1.0 SPAUC; L=2; C=3; F1=0.05; F2=0.05
Parâmetros da rodada	G=150; P=100; PC=0.85; PM=variável; T=15

Os resultados estão a seguir:

Tabela 55 : Resultados da mutação dependente da profundidade

Rodada	Erro magnético	Erro condutivo
configuração A	0.42	164.0
idem	0.50	346.5
idem	0.35	313.2
Média	**0.42**	**274.5**
configuração B	0.46	228.1
idem	0.38	225.7
idem	0.38	200.9
Média	**0.40**	**218.2**

É de se notar a regularidade apresentada por estes testes, principalmente no segundo caso. É uma das menores variações que se encontra em todo o trabalho. Entretanto, não se encontrou evidência de um ganho significativo em relação ao processamento uniforme de 3% como taxa de mutação para todas as profundidades. Quando comparado com o piso de 0.29 para erro magnético e de 125.0 para erro condutivo, estes resultados são piores, mas a quantidade de testes em ambas as configurações é diferente, pelo que a comparação fica prejudicada.

5.6.3 Comportamento do erro condutivo

Tomou-se uma rodada ao acaso e, para esta, a cada 10 gerações, o erro condutivo foi computado. O objetivo deste estudo é o de determinar eventual critério prévio de parada, fugindo-se ao critério padrão empregado neste estudo que é o de 150 gerações. Usaram-se 3 rodadas, as de número 202, 203 e 241.

Tabela 56 : Comparação entre erros condutivo e magnético ao longo das gerações

Ger	Rodada 202		Rodada 203		Rodada 241	
	Ec	Em	Ec	Em	Ec	Em
1	357.2	7.76	364.7	6.87	310.1	8.00
2	431.1	6.32	384.3	6.11	371.8	6.17
3	191.6	5.19	240.4	4.50	335.6	5.94
4	549.1	4.78	268.1	4.34	372.8	5.52
5	476.3	3.74	179.0	4.27	320.0	4.89
6	547.3	3.74	83.8	4.21	198.0	3.92
7	446.4	3.74	66.2	3.55	239.1	3.51
8	440.3	3.69	239.5	3.49	323.8	2.69
9	383.8	3.44	282.6	3.30	257.0	2.43
10	442.8	2.86	308.6	2.59	229.3	2.43
11	562.9	1.49	370.8	2.30	102.7	2.03
12	512.4	1.30	370.8	2.14	178.2	1.92
13	532.6	0.86	372.9	2.14	178.2	1.91
14	372.1	0.68	388.4	1.69	250.3	1.76
15	274.4	0.62	371.7	1.54	248.4	1.75
16	156.2	0.47	376.1	1.44	178.5	1.66
17	179.6	0.47	376.3	1.44	337.6	1.63
18	107.6	0.46	190.5	1.40	232.8	1.61
19	108.3	0.39	225.7	1.40	332.0	1.53
20	108.6	0.39	211.1	1.39	257.0	1.46
21	73.6	0.26	157.2	1.17	171.5	1.29
22	73.6	0.24	156.0	1.16	171.5	1.29
23	73.6	0.24	156.0	1.15	100.2	1.29
24	45.6	0.24	160.6	1.12	101.2	1.28
25	45.6	0.24	160.6	1.12	101.6	1.25
26	45.6	0.24	160.6	1.11	101.5	1.25
27	45.8	0.24	148.6	1.11	111.7	1.25
28	46.3	0.24	143.0	1.09	99.4	1.13
29	46.4	0.24	142.4	1.07	102.2	1.08
30	46.4	0.24	113.0	1.07	100.1	1.08
31	58.0	0.19	123.1	0.80	106.8	0.77
32	58.0	0.19	123.1	0.79	106.8	0.77
33	58.0	0.19	123.1	0.79	106.8	0.75
34	58.0	0.19	123.1	0.79	107.0	0.75
35	58.0	0.19	123.1	0.79	100.8	0.75
36	58.0	0.19	123.1	0.79	100.8	0.74
37	58.4	0.19	123.1	0.79	96.5	0.74
38	58.1	0.19	124.0	0.79	96.4	0.74
39	58.1	0.19	116.3	0.79	96.4	0.72
40	58.1	0.19	116.3	0.78	97.0	0.72
41	59.6	0.19	118.8	0.66	100.0	0.57

42	59.6	0.19	118.8	0.66	100.0	0.56
43	60.3	0.19	118.7	0.66	100.0	0.54
44	60.3	0.19	119.2	0.66	100.0	0.54
45	60.3	0.19	120.3	0.66	98.1	0.54
46	60.3	0.19	121.2	0.66	98.1	0.54
47	60.3	0.19	122.2	0.66	95.9	0.54
48	60.3	0.19	122.2	0.65	95.9	0.54
49	60.3	0.19	122.2	0.65	95.9	0.54
50	60.3	0.19	122.2	0.65	95.9	0.54
51	63.4	0.19	122.9	0.56	94.7	0.46
52	63.4	0.19	122.9	0.56	97.4	0.46
53	63.4	0.19	123.8	0.56	97.4	0.46
54	64.0	0.19	123.8	0.56	97.4	0.46
55	63.4	0.19	123.8	0.56	97.4	0.46
56	64.4	0.19	122.0	0.56	96.0	0.46
57	65.0	0.19	123.0	0.56	97.1	0.46
58	65.0	0.19	123.0	0.56	98.3	0.45
59	65.0	0.19	123.0	0.56	101.5	0.43
60	65.0	0.19	123.1	0.56	100.6	0.43
61	64.6	0.19	128.7	0.52	99.3	0.39
62	65.6	0.19	128.7	0.52	99.3	0.39
63	65.1	0.19	126.9	0.52	99.3	0.39
64	65.1	0.19	126.9	0.52	99.3	0.39
65	65.1	0.19	126.9	0.52	88.7	0.38
66	65.8	0.19	126.9	0.52	89.8	0.38
67	65.5	0.19	126.9	0.52	89.1	0.38
68	65.5	0.19	126.9	0.52	89.1	0.38
69	65.5	0.19	126.9	0.52	81.0	0.38
70	65.5	0.19	126.9	0.52	84.5	0.38
71	71.8	0.19	131.0	0.49	83.7	0.35
72	71.8	0.19	131.0	0.49	83.7	0.35
73	71.8	0.19	131.0	0.49	91.0	0.35
74	71.8	0.19	131.0	0.49	91.2	0.35
75	71.8	0.19	131.0	0.49	93.6	0.35
76	70.8	0.19	131.0	0.49	91.3	0.35
77	70.8	0.19	131.0	0.49	91.1	0.34
78	70.8	0.19	131.0	0.49	91.1	0.34
79	70.8	0.19	131.0	0.49	91.1	0.34
80	70.8	0.19	131.0	0.49	91.1	0.34
81	71.0	0.19	134.2	0.46	91.1	0.34
82	70.7	0.19	134.2	0.46	88.0	0.34
83	70.7	0.19	134.2	0.46	88.0	0.34
84	70.7	0.19	134.2	0.46	88.0	0.34
85	70.7	0.19	134.5	0.46	88.0	0.34
86	70.7	0.19	134.5	0.46	88.5	0.34
87	70.9	0.19	134.5	0.46	88.5	0.34
88	70.9	0.19	134.5	0.46	88.5	0.34
89	70.9	0.19	137.7	0.46	88.5	0.34
90	70.6	0.19	137.7	0.46	88.5	0.34
91	70.9	0.19	137.7	0.45	90.6	0.34
92	70.9	0.19	137.3	0.45	90.6	0.34
93	70.9	0.19	137.4	0.45	90.6	0.34
94	70.9	0.19	137.4	0.45	90.6	0.34
95	70.9	0.19	137.4	0.45	90.6	0.34
96	70.9	0.19	137.4	0.45	90.6	0.34
97	70.9	0.19	137.4	0.45	90.6	0.34

98	70.9	0.19	137.5	0.45	90.6	0.34
99	70.9	0.19	137.5	0.45	90.6	0.34
100	70.9	0.19	137.5	0.45	90.6	0.34
101	70.9	0.19	137.3	0.44	90.6	0.32
102	70.9	0.19	137.2	0.44	90.6	0.32
103	70.9	0.19	137.2	0.44	90.6	0.32
104	70.9	0.19	137.2	0.44	90.6	0.32
105	70.9	0.19	137.2	0.44	90.6	0.32
106	70.9	0.19	137.2	0.44	90.6	0.32
107	70.9	0.19	137.2	0.44	90.6	0.32
108	70.9	0.19	137.2	0.44	90.6	0.32
109	70.9	0.19	137.2	0.44	90.6	0.32
110	70.9	0.19	137.2	0.44	90.6	0.32
111	80.7	0.19	137.2	0.44	97.9	0.32
112	80.7	0.19	140.7	0.44	97.9	0.32
113	80.7	0.19	137.6	0.44	92.5	0.32
114	80.7	0.19	137.6	0.44	92.5	0.32
115	79.6	0.19	138.1	0.43	96.0	0.32
116	79.2	0.19	138.1	0.43	98.8	0.32
117	79.2	0.19	137.4	0.43	96.9	0.32
118	79.2	0.19	139.1	0.43	96.9	0.32
119	79.3	0.19	139.1	0.43	96.9	0.32
120	79.3	0.19	139.1	0.43	96.9	0.32
121	81.5	0.19	139.6	0.43	96.3	0.32
122	80.3	0.19	139.6	0.43	96.9	0.32
123	80.3	0.19	139.9	0.43	85.1	0.32
124	79.8	0.19	139.9	0.43	85.1	0.32
125	80.1	0.19	139.9	0.43	85.1	0.32
126	80.1	0.19	139.5	0.43	85.1	0.32
127	79.7	0.19	139.8	0.43	85.1	0.32
128	79.7	0.19	139.8	0.43	83.8	0.32
129	79.5	0.19	139.8	0.43	83.8	0.32
130	79.5	0.19	139.8	0.43	85.2	0.32
131	79.2	0.19	146.0	0.43	85.7	0.32
132	79.2	0.19	146.0	0.43	85.7	0.32
133	79.2	0.19	146.0	0.43	85.7	0.32
134	79.2	0.19	146.0	0.43	85.7	0.32
135	79.2	0.19	146.0	0.43	85.5	0.32
136	79.2	0.19	146.1	0.43	85.5	0.32
137	79.2	0.19	146.1	0.43	85.5	0.32
138	79.2	0.19	146.1	0.42	85.5	0.32
139	79.2	0.19	146.1	0.42	85.5	0.32
140	79.2	0.19	146.1	0.42	85.5	0.32
141	79.2	0.19	146.3	0.42	84.7	0.32
142	79.2	0.19	146.3	0.42	84.3	0.32
143	79.5	0.19	146.3	0.41	85.5	0.32
144	79.5	0.19	146.3	0.41	85.5	0.32
145	79.5	0.19	146.3	0.41	85.1	0.32
146	79.5	0.19	146.3	0.41	85.1	0.32
147	79.5	0.19	146.3	0.41	85.1	0.32
148	79.5	0.19	146.3	0.41	85.1	0.32
149	79.5	0.19	146.3	0.41	85.1	0.32

De modo gráfico, eis o comportamento dos erros condutivos, para estas três rodadas

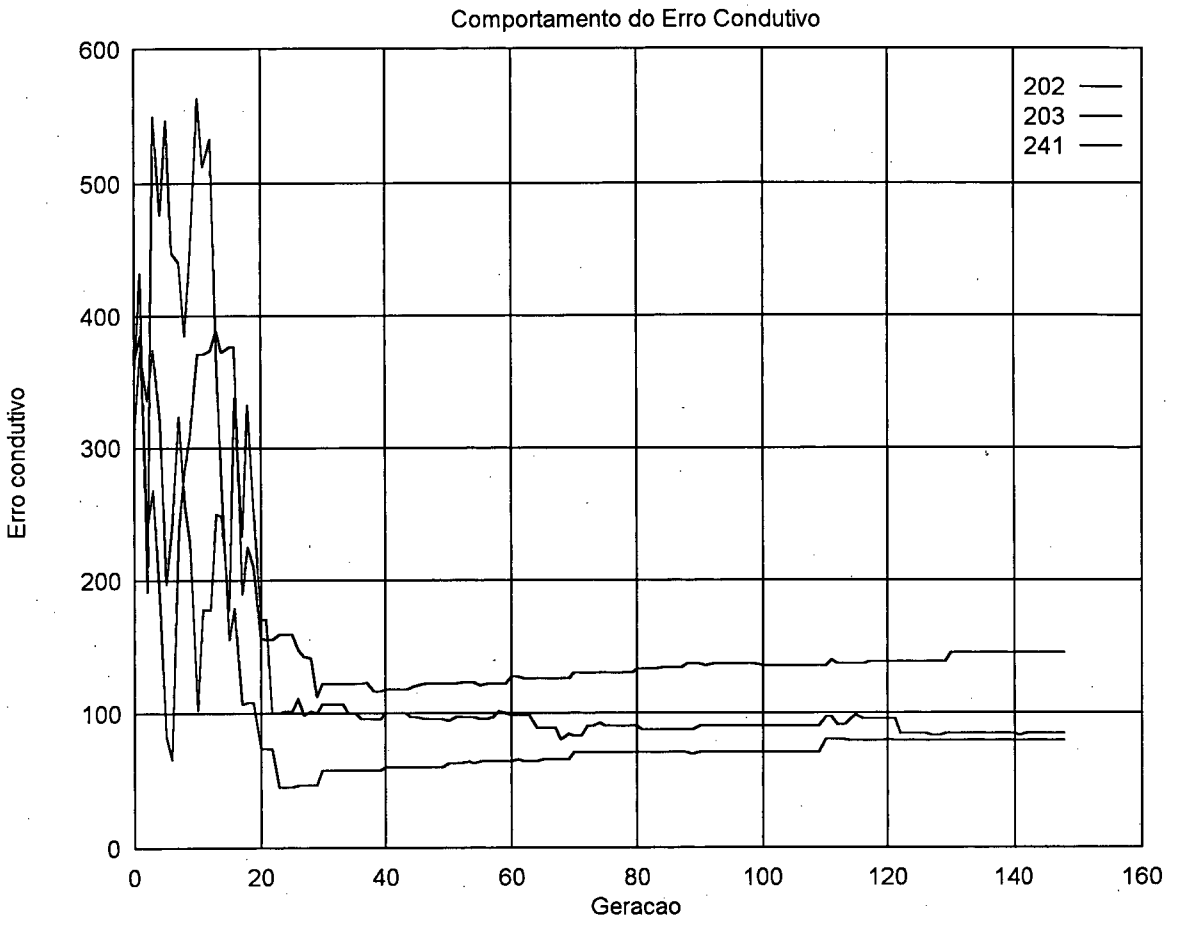


Fig. 34: Comportamento do erro condutivo ao longo das gerações

Ao se observar apenas para o erro condutivo, percebe-se que a busca evolutiva poderia parar antes das 150 gerações. Em duas das execuções mostradas, após 1/3 do curso completo o erro evolutivo cresceu. Entretanto, há que se recordar que o erro evolutivo aqui só é conhecido porque se trata de dados sintéticos. Em situações em que os dados sejam reais, o erro condutivo é desconhecido e apenas se tem o erro magnético (que recordando é o único que guia a busca evolutiva). Em se tratando deste segundo erro (o magnético), o critério de 150 gerações parece adequado, pois nos três casos ele permite a estabilização do erro magnético.

5.6.4 Participação individual de cada um dos operadores

O teste agora consiste em analisar o desempenho do engenho evolutivo. Usou-se a configuração número 1, e, para cada teste, efetuaram-se 2 rodadas com sementes aleatórias distintas. Eis o teste:

Tabela 57: Parâmetros dos testes de mutação dependente da profundidade

Parâmetro	Valor

Configuração utilizada	1
Distribuição da população inicial	100% igual a 10
Ruído	1; A=3; T=médio; R=0.01
Otimização externa	1 a cada 10 gerações; local=5+gen/10;homo=3
Elitismo	1
Crossover	1.0 SPAUC; L=2; C=3; F1=0.05; F2=0.05
Parâmetros da rodada	G=150; P=100; PC=0.85; PM=variável; T=15

Os resultados obtidos foram:

Tabela 58 : Resultados da eliminação individual de um único operador

Rodada	Erro magnético	Erro condutivo
rodada completa	0.30	114.6
idem	0.37	324.1
Média	0.34	219.3
sem busca local	0.64	93.7
idem	0.96	184.5
Média	0.80	139.1
sem SPAUC	0.55	241.9
idem	0.38	225.7
Média	0.46	233.8
sem homogeneização	0.42	278.2
idem	0.44	402.8
Média	0.43	340.5

Ambos os resultados do erro magnético da rodada completa são melhores que quaisquer outros. Isso parece mostrar a adequação do trabalho conjunto dos 3 operadores. Note-se que quando se deixa a busca local de fora, o erro condutivo apresenta resultados melhores do que o conjunto dos 3 operadores. Este resultado deve ser interpretado como uma anomalia, de resto só conhecida a posteriori. A anomalia está caracterizada pela baixa correlação entre erro magnético e condutivo que surgiu nesta execução devido ao caráter estocástico do processo.

5.6.5 Estudo de dispersão do resultado

O teste agora visa determinar qual a dispersão dos indivíduos da população após uma rodada padrão.

Usou-se uma rodada escolhida ao acaso, segundo o seguinte esquema:

Tabela 59 : Parâmetros dos testes de dispersão dos indivíduos na população

Parâmetro	Valor
Configuração utilizada	1
Distribuição da população inicial	100% igual a 10

Ruído	1; A=3; T=médio; R=0.01
Otimização externa	1 a cada 10 gerações; local=5+gen/10;homo=3
Elitismo	1
Crossover	1.0 SPAUC; L=2; C=3; F1=0.05; F2=0.05
Parâmetros da rodada	G=150; P=100; PC=0.85; PM=variável; T=15

Os resultados desta rodada, acima descrita, estão a seguir. Como sempre apresenta-se o desempenho do melhor indivíduo da população ao final do processo evolutivo.

Tabela 60 : Resultados da dispersão dos indivíduos da população

Rodada	Erro magnético	Erro condutivo
rodada completa	0.30	114.3

Esta rodada apresentou os seguintes valores para erro condutivo para os 100 indivíduos ao final de 150 gerações:

Tabela 61 : Erros condutivos de toda a população após 150 gerações

281.94	162.12	267.82	129.25	165.11	245.02	114.28	121.61	167.82	157.82
197.56	143.51	208.22	287.63	265.24	202.09	266.56	186.38	197.58	239.56
270.73	303.15	414.34	246.45	227.46	114.58	187.56	196.66	285.97	186.96
169.35	201.05	242.30	243.95	114.57	114.57	114.28	277.99	214.67	193.44
307.36	274.08	195.53	142.52	127.45	217.43	312.87	160.33	232.10	191.52
187.34	439.84	116.71	260.95	132.22	169.34	128.63	185.79	114.57	129.24
141.88	230.35	114.56	277.64	294.42	276.13	367.77	209.65	319.55	172.03
202.14	204.28	151.92	453.88	216.80	124.41	172.47	201.74	114.28	173.47
216.29	308.47	328.96	198.95	243.47	114.57	205.66	136.61	263.37	232.78
247.40	120.22	170.15	255.13	114.54	245.94	165.76	134.05	136.64	177.89

A média desta tabela é de 207.89, o maior valor é de 453.88 e o menor é de 114.27.
O desvio padrão é de 73.07 e a variância é de 5339.48. Se, ao invés de se trabalhar apenas com o melhor resultado (114.2790327), houvesse a opção pela média dos 10 melhores resultados, esta seria de 114.4791068. A média dos 20 melhores é de 120.4294199. A média dos 50 melhores é de 151.0332092.

6. Resultados para os demais problemas

Neste capítulo são apresentados os resultados dos problemas 2 e 3, denominados transmissão de calor e condutividade térmica. O grau de detalhe obtido nestes resultados é menor do que aquele obtido no problema 1, uma vez que este foi o problema escolhido para a elaboração da sistemática dos operadores específicos e os outros dois problemas serviram apenas para experimentar e validar as conclusões obtidas no problema 1.

6.1 Para o problema de transmissão de calor

Para o estudo da solução deste problema, optou-se por adaptar as mesmas ferramentas metodológicas usadas no problema anterior. O objetivo de assim se agir foi o de consolidar e generalizar as conclusões lá obtidas.

A placa de material composto é uma grade de 9 x 9 partes, baseado na similariedade anteriormente descrita (o problema anterior tem 7 x 10 = 70 componentes, e este tem que tê-las em quantidade tal que seja resultado de um número ímpar elevado ao quadrado. Optou-se por 9, que ao quadrado dá 81 que é próximo a 70). Dois materiais distintos podem compor a placa. O primeiro, denominado material base tem condutividade térmica de 10,0 W m^{-1} K^{-1} e estará representado nos diagramas a seguir por uma célula branca. Já o material de preenchimento, tem condutividade térmica de 0,1 W m^{-1} K^{-1} e estará representado por células de cor cinza, nas figuras a seguir.

6.1.1 Casos de Estudo

Buscando similaridade com o caso anterior, criaram-se 4 casos de estudo:

6.1.1.1 Caso 1

Esta placa é formada por 81 "casas" sendo 73 do material base e apenas 8 do material de preenchimento. Ela busca similaridade com o caso 1 do problema 1, apenas com a diferença de que lá as duas manchas eram de valores menor e maior do que o valor do material restante. Aqui isso não é possível.

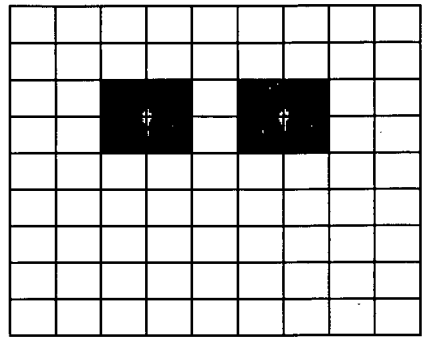


Fig. 35: Caso 1 para a placa de material composto

A condutividade térmica para esta configuração, denominada "caso 1" é de 8.430446870150 W m^{-1} K^{-1}.

6.1.1.2 Caso 2

Esta placa tem 10 casas do material de preenchimento. A similaridade com o caso equivalente do problema 1 é quase total.

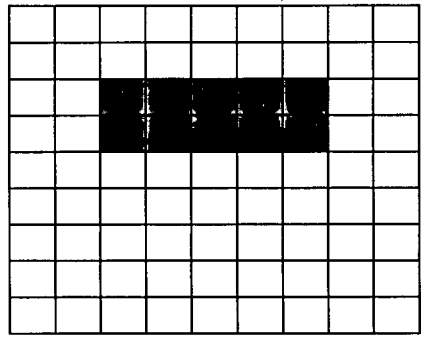


Fig. 36: Caso 2 para a placa de material composto

A condutividade térmica deste conjunto (caso 2) é de 8.331995172741 W m^{-1} K^{-1}.

6.1.1.3 Caso 3

Este caso repete a mesma disposição do elemento de preenchimento verificado no caso equivalente do problema 1. Possui 19 casas de material de preenchimento.

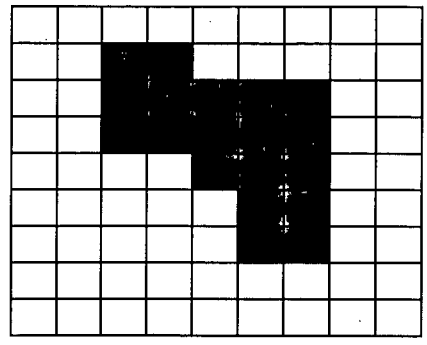


Fig. 37: Caso 3 para a placa de material composto

A condutividade térmica para este conjunto 3 é de 5.170600787843 W m^{-1} K^{-1}.

6.1.1.4 Caso 4

Este conjunto também repete com bastante similaridade o caso equivalente do problema 1. Trata-se de uma única casa de material de preenchimento colocada no topo e no centro da placa.

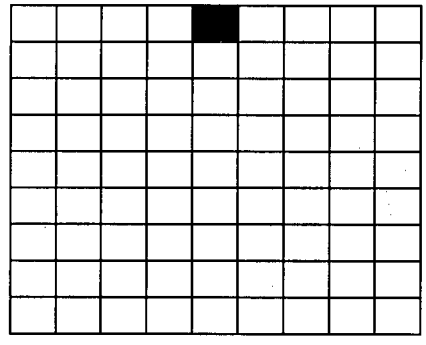


Fig. 38: Caso 4 para a placa de material composto

A condutividade térmica deste caso número 4 é de 9.697087397105 W m^{-1} K^{-1}.

6.1.2 Operadores e Função Objetivo Usados

Dos 3 operadores anteriormente desenvolvidos (SPAUC, homogeneização e busca local), apenas a busca local não foi implementada. O SPAUC utilizado foi rigorosamente o mesmo do problema anterior, bem como a homogeneização.

Em uma primeira abordagem utilizou-se a função objetivo similar também a do problema anterior, a saber:

$$f_1 = \frac{\alpha_1}{(1+0.1)^{|\varepsilon - R|}}$$

onde ε é a condutividade térmica equivalente de um determinado indivíduo da solução e R é a condutividade térmica do conjunto que se busca e que é conhecido anteriormente. Surgiu aqui, um inesperado problema. O que determina o valor da condutividade equivalente é a quantidade de placas de material de preenchimento e não a sua disposição na grade da solução. Em 7 experimentos, usando-se apenas esta parcela na função objetivo, o engenho evolutivo convergiu para uma configuração qualquer, desde que respeitada a quantidade total de partes do material de preenchimento. Para forçar o surgimento da solução esperada (na qual, os materiais de preenchimento se aglutinam de alguma maneira, possivelmente por restrições de ordem tecnológica na construção da placa), houve que introduzir novas parcelas na função objetivo. Entretanto tais parcelas diminuiram artificialmente o espaço de busca e é possível discutir-se se é lícita esta abordagem.

Como exemplo do aqui afirmado, tome-se uma execução que busca reconstituir a configuração 1, levando em consideração f1 (ou seja, $\alpha 1 \neq 0$).

Após 150 gerações, o melhor indivíduo é o figura 39, cujo valor da função objetivo é de 0.9999661. A condutividade térmica desta solução é de 8.430090809730 W m^{-1} K^{-1}. Compare-se com o valor padrão buscado que é de 8.430446870150 W m^{-1} K^{-1}. Note-se que a solução tem 8 placas de material inserido, tal como o padrão, mas estas se encontram "espalhadas".

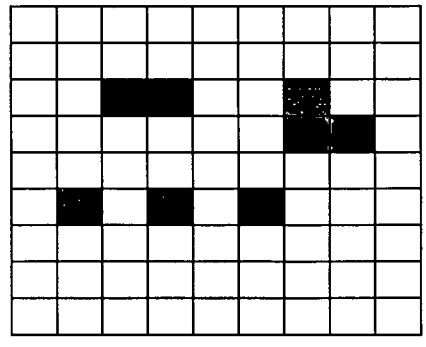


Fig. 39: Problema 2, Caso 1: Melhor indivíduo após 150 gerações

6.1.3 Solução do Problema

As configurações 1, 2 e 3 não apresentaram soluções satisfatórias. O número de blocos era correto, mas não sua disposição espacial. O único caso bem sucedido foi o da configuração 4. Este teve 70% dos valores de todos os indivíduos inicializados com 0 (condutividade de 10,0 W m^{-1} K^{-1}). Usou-se elitismo, a recombinação foi o SPAUC em 100% dos casos. Foram 150 gerações com populações de 100 indivíduos, probabilidade de crossover igual a 0.85, probabilidade de mutação de 0,03, tamanho de torneio de 15 indivíduos. A tabela 62 resume os parâmetros utilizados no teste.

Tabela 62 Resumo dos parâmetros da configuração 4 para o problema do calor

Parâmetro	Valor
Configuração utilizada	4
Distribuição da população inicial	70% igual a 0
Ruído	sem ruído
Otimização externa	15 tentativas de homogeneização por geração
Elitismo	1
Crossover	1.0 SPAUC; L=2; C=3; F1=0.05; F2=0.05
Parâmetros da rodada	G=150; P=100; PC=0.85; PM=0.03; T=15; SA=0.1987
Parâmetros da função objetivo	α_1 = 1.0

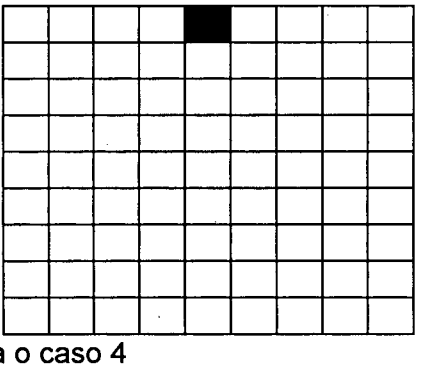


Fig. 40: Melhor indivíduo para o caso 4

O resultado alcançado foi o descrito na figura 40, que foi encontrado na 33ª geração. A condutividade equivalente é de 9.697087397105 W m^{-1} K^{-1} e a função objetivo é de 1.0000000. Apenas o caso 4 foi resolvido, devido à simplicidade da distribuição de materiais na placa.

Como conclusão, pode-se afirmar que o uso de um ferramental evolutivo munido de "pouco" conhecimento do problema 2 resultou insuficiente para sua solução, ao contrário do que ocorreu no problema 1. A explicação para este fato parece estar no fato de que configurações completamente distintas, mas que compartilhem o mesmo número de

casas de preenchimento compartilham igualmente resultado semelhante até a 3ª ou 4ª casa decimal para o valor da função objetivo pura, como por exemplo *f1* acima citada.

Está-se diante de uma situação em que a função objetivo tem característica de alta multimodalidade. Tal situação apresenta dificuldades para a solução do problema seja qual for o otimizador utilizado. O engenho evolutivo não fugiu a esta asserção como se pôde ver acima.

6.2 Para o problema de Difusão de Calor em Materiais Compostos

Segue-se a descrição dos resultados obtidos para este problema. Na medida do possível, manteve-se o mesmo arcabouço de ferramentas utilizados nos problemas anteriores a fim de obter resultados comparáveis e permitindo consolidar o uso e validação das técnicas.

6.2.1 Caso 1D

Trata-se de uma disposição de uma peça em uma dimensão, composta por 10 camadas homogêneas, cada uma com uma espessura de 0,3mm. Logo após o início da experiência o conjunto é submetido a um pulso térmico com duração de 1.6 seg e com densidade de fluxo de 4500 W/m^2. As diferentes camadas são separadas pelas suas resistências térmicas de contacto. Para efeito de comparação com os resultados apresentados para este problema em Ramos (1992) usaram-se as mesmas situações de teste, descritas na tabela 63.

Tabela 63 - Parâmetros da reconstrução 1D (extraído de Ramos, 1992)

Parâmetro	Valor
geometria	peça multicamada em 1 dimensão de 10 x 0.3mm
Propriedades termofísicas	α=2.2x10^{-7} m^2/s, k = 0.67 W/m/K, h=10 W/m^2/K
impulsão térmica	τ=1.6 s, q_{flux} = 4500 W/m^2
modelo direto	Δx=0.15mm, Δt=0.16s t_{total}=20 s e N = 40
modelo inverso	número de parâmetros a estimar = 9
constantes de inversão	u_q = 1 x 10^{-2} e l_q = 1 x 10^{-8}

Para o engenho evolutivo, utilizou-se um cromossomo de 9 valores. Como o que buscava era o coeficiente térmico de contato, na forma de um expoente para a base 10, adotou-se o mesmo intervalo de variação da apresentação original e que é:

limite mínimo de coeficiente térmico de contato: 10^{-8} W/m^2/K

limite máximo de coeficiente térmico de contacto: 10^{-2} W/m^2/K

O que se busca encontrar é o vetor $p \equiv \{p_q = r^{-1}_q, q=1, ..., 9\}$. Portanto, p variará entre 2 e 8. No engenho evolutivo utilizou-se um tamanho de alfabeto de 7.

A configuração buscada para os coeficientes térmicos de contacto foi a de

10^{-8}	10^{-3}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}

Que traduzida para o cromossomo, já com os valores de p, tal como aqui codificado, ficou

8	3	8	8	8	8	8	8	8

Definiu-se o resíduo R(p) que vem a ser a somatória das diferenças quadráticas entre as temperaturas obtidas pela aplicação do modelo direto a $p_{q.ex}$ e aquelas obtidas quando o modelo direto é aplicado a p_q.

Os testes aplicados encontram-se descritos na tabela 64.

Tabela 64 : Problema de difusão de calor: testes comparativos entre Ramos (1992) e abordagem evolutiva

Teste	Vetor p buscado	Característica
1	(8, 3, 3, 3, 8, 8, 8, 8, 8)	Inicialização igual ao limite inferior (p_i=8) e sem ruído
2	(8, 3, 3, 3, 8, 8, 8, 8, 8)	Inicialização igual a p=(2, 8, 8, 8, 8, 8, 8, 8, 2), sem ruído
3	(8, 3, 3, 3, 8, 8, 8, 8, 8)	Inicialização igual ao limite superior (p_i=2), sem ruído
4	(8, 8, 8, 3, 8, 8, 8, 8, 8)	Inicialização igual ao limite inferior sem ruído
5	(8, 8, 8, 3, 8, 8, 8, 8, 8)	Inicialização igual ao limite inferior e com ruído de variância 0.025° C.
6	(8, 8, 8, 3, 8, 8, 8, 8, 8)	Inicialização igual ao limite inferior e com ruído de variância 0.1° C.

A coluna "Vetor p buscado" deve ser entendida como sendo os expoentes de uma base constante igual a 10. Os expoentes estão com o sinal trocado. Em outras palavras, a configuração primeira (8, 3, 3, 3, 8, 8, 8, 8, 8) deve ser entendida como (10^{-8}, 10^{-3}, 10^{-3}, 10^{-3}, 10^{-8}, 10^{-8}, 10^{-8}, 10^{-8}, 10^{-8}).

Para comparação posterior, os resultados para estes 6 testes apresentados em Ramos (1992) são:

Tabela 65 : Resultados em Ramos (1992)				
T.	Vetor p buscado	Vetor p encontrado	na iteração	Resíduo quadrático
1	(8, 3, 3, 3, 8, 8, 8, 8, 8)	o mesmo p buscado	30	0.0
2	(8, 3, 3, 3, 8, 8, 8, 8, 8)	o mesmo p buscado	43	0.0
3	(8, 3, 3, 3, 8, 8, 8, 8, 8)	o mesmo p buscado	70	0.0
4	(8, 8, 8, 3, 8, 8, 8, 8, 8)	o mesmo p buscado	20	0.0
5	(8, 8, 8, 3, 8, 8, 8, 8, 8)	(10^{-8}, 10^{-8}, 10^{-8}, 8.89^{-4}, 1.37^{-4}, 10^{-8}, 10^{-8}, 10^{-8}, 10^{-8})	8	7.45E-02
6	(8, 8, 8, 3, 8, 8, 8, 8, 8)	(10^{-8}, 10^{-8}, 2.1^{-5}, 9.24^{-4}, 10^{-8}, 10^{-8}, 10^{-8}, 3.14^{-4}, 10^{-8})	18	1.03E+00

O engenho evolutivo atuando sobre os mesmos casos e com os parâmetros da tabela 66, apresentou o desempenho discriminado na tabela 67.

Tabela 66 : Problema de difusão de calor: parâmetros dos testes para o caso 1D	
Parâmetro	Valor
Otimização externa	aplicada 18 vezes (9 para cima e 9 para baixo) a cada 10 gerações.
Elitismo	1
Crossover	uniforme, pois este caso é 1D
Parâmetros da rodada	G=variável; P=100; PC=0.85; PM=0.03; T=4

Tabela 67 : Problema de difusão de calor: resultados do Engenho Evolutivo					
T.	Vetor p buscado	Vetor p encontrado	Geração/número de chamadas à f objetivo	Resíduo quadrático	Semente aleatória
1A	(8, 3, 3, 3, 8, 8, 8, 8, 8)	o mesmo p buscado	5 / 468	0.0	0.555
1B	(8, 3, 3, 3, 8, 8, 8, 8, 8)	o mesmo p buscado	6 / 594	0.0	0.777
1C	(8, 3, 3, 3, 8, 8, 8, 8, 8)	o mesmo p buscado	12 / 1253	0.0	0.999
2A	(8, 3, 3, 3, 8, 8, 8, 8, 8)	o mesmo p buscado	18 / 1733	0.0	0.555
2B	(8, 3, 3, 3, 8, 8, 8, 8, 8)	o mesmo p buscado	27 / 2628	0.0	0.777
2C	(8, 3, 3, 3, 8, 8, 8, 8, 8)	o mesmo p buscado	27 / 2615	0.0	0.999
3A	(8, 3, 3, 3, 8, 8, 8, 8, 8)	o mesmo p buscado	19 / 1947	0.0	0.555
3B	(8, 3, 3, 3, 8, 8, 8, 8, 8)	(5, 3, 3, 4, 3, 4, 3, 8, 8)	50 / 5100	2.1E-01	0.777
3C	(8, 3, 3, 3, 8, 8, 8, 8, 8)	o mesmo p buscado	19 / 1899	0.0	0.999
4A	(8, 8, 8, 3, 8, 8, 8, 8, 8)	o mesmo p buscado	2 / 212	0.0	0.555
4B	(8, 8, 8, 3, 8, 8, 8, 8, 8)	o mesmo p buscado	1 / 107	0.0	0.777
4C	(8, 8, 8, 3, 8, 8, 8, 8, 8)	o mesmo p buscado	2 / 202	0.0	0.999
5A	(8, 8, 8, 3, 8, 8, 8, 8, 8)	o mesmo p buscado	0 / 25	2.71E-01	0.555
5B	(8, 8, 8, 3, 8, 8, 8, 8, 8)	o mesmo p buscado	0 / 7	2.71E-01	0.777
5C	(8, 8, 8, 3, 8, 8, 8, 8, 8)	o mesmo p buscado	2 / 198	2.70E-01	0.999
6A	(8, 8, 8, 3, 8, 8, 8, 8, 8)	o mesmo p buscado	2 / 220	1.07E+00	0.555

6B	(8, 8, 8, 3, 8, 8, 8, 8, 8)	o mesmo p buscado	0 / 7	1.08E+00	0.777
6C	(8, 8, 8, 3, 8, 8, 8, 8, 8)	o mesmo p buscado	2 / 238	1.09E+00	0.999

As conclusões que se podem obter quando se comparam os dois métodos de solução, podem ser assim estabelecidas:

Quando a inicialização é muito próxima ao vetor buscado, e não há ruído, o método descrito em Ramos (1992) encontrou o resultado na iteração 30, tendo efetuado um mínimo de 540 chamadas à função objetivo. O método evolutivo encontro resultado equivalente com uma média de 771 chamadas. No melhor caso evolutivo o número de chamadas foi de 468. Note-se que para o método evolutivo devem ser efetuadas várias medidas em cada caso e a média deve ser tomada (trata-se de um método estocástico). Já o método de Ramos (1992) é determinístico e por conseqüência basta uma execução. Finalmente, tanto o resíduo R(p) quanto o erro ε são iguais a zero, o que caracteriza 100% de acerto em ambos os métodos.

A segunda bateria de testes considerou uma inicialização igual a (2, 8, 8, 8, 8, 8, 8, 8, 2) e ausência de ruído. No método descrito em Ramos (1992), o resultado buscado foi encontrado na geração 43, com um mínimo de 774 chamadas à função objetivo. O engenho evolutivo encontrou um resultado de mesma qualidade (R(p) e ε iguais a zero) com 2325 chamadas à função objetivo, em média e com melhor resultado de 1733 chamadas.

A terceira bateria, teve inicialização igual a (2, 2, 2, 2, 2, 2, 2, 2, 2) ou seja quase o extremo oposto às demais inicializações até agora usadas e 0% de ruído. O método de Ramos (1992) encontrou resultado de $R(p) = \varepsilon = 0.0$ em 70 gerações com um número mínimo de 1260 chamadas à função objetivo enquanto que o engenho evolutivo somente encontrou este resultado em 2 das 3 execuções feitas. Nestas a média de chamadas foi de 1923.

No quarto caso, o resultado buscado continha apenas um único valor diferente do padrão de inicialização e ausência de ruído. Ambos os métodos encontraram o resultado correto ($R(p) = \varepsilon = 0$). Ramos (1992) apresenta o resultado com 20 iterações (mínimo de 360 chamadas à função objetivo) enquanto que o engenho evolutivo chegou ao mesmo resultado com média de 173 chamadas.

No quinto caso, adicionou-se um ruído gaussiano com variância de 0,025° C. Como era de se esperar, este fato impediu ambos os métodos de encontrar o valor zero para o resíduo. Ramos (1992) apresenta um resíduo de 0,0745 na iteração 8 (mínimo de

144 chamadas) enquanto que o engenho evolutivo apresentou resíduo de 0,271, com média de 76 chamadas à função objetivo.

Finalmente, no último caso o vetor buscado foi (8, 8, 8, 3, 8, 8, 8, 8, 8) e a inicialização foi (8,8, 8, 8, 8, 8, 8, 8, 8). O ruído adicionado teve variância igual a 0,1° C. Ramos (1992) apresenta como resultado um resíduo de 1.03 na iteração 18 (mínimo de 324 chamadas). O engenho evolutivo encontrou um resíduo de 1.08 (na média) e 155 chamadas à função objetivo (na média).

Concluindo, pode-se afirmar que para o caso 1D, o método evolutivo apresenta desempenho satisfatório tanto quando comparado em termos de consumo de recursos (chamadas à função objetivo), como quando comparado quanto à qualidade do resultado (valores de R(p) e ε). A adição de ruído não compromete os resultados do engenho evolutivo o que caracteriza sua robustez quanto a esta condição.

6.2.2 Caso 2D

Para este caso, utilizaram-se os seguintes parâmetros

Tabela 68 Parâmetros da reconstrução 2D (extraído de Ramos, 1992)

Parâmetro	Valor
geometria	2D
Propriedades termofísicas	α_{rad}=2.2x10^{-6} m^2/s, α_{axi}=2.2 x 10^{-7} m^2/s, k_{rad} = 6.7 W/m/K k_{axi} = 0.67 W/m/K, h=10 W/m^2/K
impulsão térmica	τ=2.0 s, q_{flux} = 15K W/m^2
modelo direto	15 x 10 nodos, Δx=0.30mm, Δy=3.00mm, Δt=0.5s t_{total}=15 s e N = 4
modelo inverso	número de parâmetros a estimar = 40, V_{min}=5, N_{obs}=300
constantes de inversão	u_q = 1 x 10^{-2} e l_q = 1 x 10^{-8}

Utilizou-se também uma medida de erro extraída de Ramos (1992) denominada erro logarítmico normalizado definido por

$$\frac{\varepsilon^r_{\log}}{\varepsilon^0_{\log}} = \left(\frac{\sum_{q=1}^{Q} \log^2 (p^r_q / p_{q.ex})}{\sum_{q=1}^{Q} \log^2 (p^0_q / p_{q.ex})} \right)^{1/2}$$

Este erro correlaciona os coeficientes térmicos da solução original (que originou ou Resíduos, após a aplicação do modelo direto) comparados com os coeficientes térmicos da solução do problema inverso. Fazendo-se analogia com o problema 1 (o de reconstrução das distribuições da condutividade geoelétrica), poderíamos ter:

Tabela 69 - Analogia entre as medidas do Problema 1 e do Problema 3		
Parâmetro comparativo	Problema 1	Problema 3
Erro direto: medida da diferença entre o resultado do modelo direto aplicado ao padrão e ao indivíduo sob estudo (caso sintético) ou diferença entre a medida de campo e a aplicação do modelo direto ao indivíduo sob estudo.	Erro magnético	Resíduo
Erro indireto: medida da diferença entre o indivíduo padrão e o em estudo	Erro condutivo	Erro logarítmico

O erro direto é o único obtenível em situações reais, não sintéticas, nas quais a solução buscada é desconhecida a *priori*.

Para o teste a seguir, usou-se uma disposição de defeitos em triângulo, cuja representação, na forma de coeficientes térmicos de contacto é:

10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-3}	10^{-3}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-8}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-8}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}

Para efeito de comparação com Ramos (1992), houve a necessidade de "desnormalizar" as medidas originais, uma vez que o denominador da fórmula vista acima introduz um erro devido ao conceito diferenciado de inicialização que é dado por ambos os métodos. Enquanto em Ramos (1992), o método é beneficiado caso o ponto inicial esteja nas proximidades do ponto buscado, nos métodos evolutivos isto não é verdade: uma inicialização aleatória (pelo menos em parte) é buscada e mais ainda, benéfica.

Os resultados, devidamente "desnormalizados", são:

Tabela 70 - Resultados de Ramos (1992)	
Parâmetro	Valor em Ramos 1992
erro logarítmico	na iteração 25, ε = 12.3326 na iteração 50, ε = 6.5647 na iteração 75, ε = 6.4430 na iteração 100, ε = 4.2609
Resíduo R(p)	na iteração 25, R(p) = 1.360 na iteração 50, R(p) = 0.806 na iteração 75, R(p) = 0.720 na iteração 100, R(p) = 0.706
tempo de CPU em segundos	na iteração 25, t=351 seg na iteração 50, t=841 seg na iteração 75, t=1310 seg na iteração 100, t=1780 seg

Para efeito de comparação, eis os resultados conseguidos pelo engenho evolutivo

Os parâmetros das rodadas evolutivas são

Tabela 71 : Problema de difusão de calor, parâmetros dos testes para o caso 2D	
Parâmetro	Valor
Otimização externa	busca local a cada 2 gerações, aplicada ao melhor indivíduo, com 40 tentativas positivas e 40 negativas. O incremento é 1.
Elitismo	1
Inicialização da população	80% igual a 1E-8
Crossover	100% de SPAUC
Parâmetros da rodada	G=variável; P=100; PC=0.85; PM=0.03; T=15

Os resultados,

Tabela 72 : Resultados do engenho evolutivo (inicialização quase homogênea)			
Parâmetro	Valor	Valor	Valor
Semente	0.111	0.222	0.333
erro logarítmico	iteração 10, ε = 13.171 iteração 20, ε = 11.818 iteração 30, ε = 9.865 iteração 40, ε = 9.479 (*)	10, ε = 11.952 20, ε = 9.844 30, ε = 10.019 40, ε = 9.746	10, ε = 17.783 20, ε = 15.582 30, ε = 14.941 40, ε = 14.913
Resíduo R(p)	iteração 10, R(p) = 1.112 iteração 20, R(p) = 0.730 iteração 30, R(p) = 0.587 iteração 40, R(p) = 0.524	10, R(p) = 1.357 20, R(p) = 1.076 30, R(p) = 1.008 40, R(p) = 0.959	10, R(p) = 2.583 20, R(p) = 1.490 30, R(p) = 1.322 40, R(p) = 1.235
tempo de CPU em segundos	iteração 10, t = 360 seg iteração 20, t = 675 seg iteração 30, t = 1410 seg iteração 40, t = 1920 seg	10, t = 360 seg 20, t = 675 seg 30, t = 1410 seg 40, t = 1920 seg	10, t = 360 seg 20, t = 675 seg 30, t = 1410 seg 40, t = 1920 seg

O melhor indivíduo nas 3 rodadas acima, identificado com (*), apresentado pelo engenho evolutivo em 40 gerações é:

3.1E-06	3.1E-06	1.0E-08	1.0E-08	9.8E-04	9.0E-04	2.9E-04	1.0E-08	1.0E-08	1.0E-08
9.2E-06	9.2E-06	1.8E-05	1.4E-05	9.8E-04	9.8E-04	3.1E-06	1.0E-08	1.0E-08	1.0E-08
3.4E-05	3.7E-05	1.8E-05	9.4E-04	9.6E-04	1.4E-03	1.6E-03	1.6E-03	1.0E-08	1.0E-08
1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08

Outro teste efetuado, agora com inicialização aleatória dos dados

Tabela 73 : Problema de difusão de calor, parâmetros dos testes para o caso 2D	
Parâmetro	Valor
Otimização externa	busca local a cada 2 gerações, aplicada ao melhor indivíduo, com 40 tentativas positivas e 40 negativas. O incremento é 5.
Elitismo	1
Inicialização da população	aleatória
Crossover	100% de SPAUC
Parâmetros da rodada	G=variável; P=100; PC=0.85; PM=0.03; T=15

Os resultados,

Tabela 74 - Resultados do engenho evolutivo (inicialização aleatória)

Parâmetro	Valor	Valor	Valor
Semente	0.111	0.222	0.333
erro logarítmico	iteração 10, ε = 17.471 iteração 20, ε = 15.093 iteração 30, ε = 13.540 iteração 40, ε = 12.340	10, ε = 18.402 20, ε = 17.280 30, ε = 13.713 40, ε = 12.177	10, ε = 18.659 20, ε = 13.172 30, ε = 9.168 40, ε = 9.123 (**)
Resíduo R(p)	iteração 10, R(p) = 1.120 iteração 20, R(p) = 0.617 iteração 30, R(p) = 0.521 iteração 40, R(p) = 0.466	10, R(p) = 2.300 20, R(p) = 1.701 30, R(p) = 1.334 40, R(p) = 1.126	10, R(p) = 2.602 20, R(p) = 1.587 30, R(p) = 1.453 40, R(p) = 1.453
tempo de CPU em segundos	iteração 10, t = 360 seg iteração 20, t = 675 seg iteração 30, t = 1410 seg iteração 40, t = 1920 seg	10, t = 360 seg 20, t = 675 seg 30, t = 1410 seg 40, t = 1920 seg	10, t = 360 seg 20, t = 675 seg 30, t = 1410 seg 40, t = 1920 seg

O melhor indivíduo nas 3 rodadas acima, identificado por (**), apresentado pelo engenho evolutivo em 40 gerações é:

9.8E-06	1.0E-08	1.0E-08	1.8E-04	1.0E-03	8.5E-04	2.2E-04	9.8E-06	6.2E-07	6.7E-06
1.6E-05	3.1E-06	1.2E-05	2.1E-04	1.0E-03	1.0E-05	3.6E-04	2.0E-04	6.2E-07	1.0E-08
1.0E-08	2.2E-05	5.4E-04	3.8E-06	6.5E-04	9.6E-06	2.4E-04	3.1E-05	1.9E-05	3.1E-06
1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08

Como conclusão para o caso 2D, verifica-se que o resultado encontrado em Ramos (1992) é de ε = 4.2609 e R(p)=0.706 em 100 iterações que demandaram um tempo de processamento em uma máquina UNIX de médio porte (de 1992, 8 anos atrás) de 1708 segundos. A quantidade mínima de chamadas à função objetivo foi de 8000 chamadas. Já para o engenho evolutivo executaram-se 6 rodadas. As três primeiras tiveram a população inicializada com o valor menor (1E-8) em 80% dos valores. As outras 3 rodadas tiveram inicialização 100% aleatória. Em ambos os casos a resolução do espaço de busca no espaço de soluções é de 0.000000610388177, considerada baixa e adequada ao problema.

Para o primeiro teste, o menor erro encontrado foi de ε = 9.479, para uma média nas três medidas de ε = 11.3793. O resíduo no melhor caso foi de R(p)=0.524 e na média R(p)=0.906. O engenho evolutivo demandou cerca de 6000 chamadas à função objetivo, gastando 1920 segundos em um PC de 500MHz. Para o caso que teve inicialização aleatória, o menor erro foi de ε = 9.123 e o menor resíduo foi de R(p)=0.466, para valores médios de ε = 11.2133 e R(p)=1.015.

No algoritmo evolutivo, percebeu-se que quando se busca otimizar ε, a inicialização aleatória tem melhor desempenho, enquanto que quando se busca otimizar o resíduo, o algoritmo inicializado com o piso dos valores (1E-8) teve o melhor desempenho. Finalmente, pôde-se ver que não há uma correlação firme entre ε e R(p). Enquanto o primeiro mede o desempenho global da solução, o segundo guia o engenho evolutivo. Como se pode ver na tabela 73, a coluna com melhor R(p) apresenta o pior ε e vice-versa. Este fato, característica dos problemas inversos, demonstra a dificuldade de sua solução.

6.2.3 Caso 3D

Para este problema está-se diante de um espaço de busca na ordem de 7^{400} ou 16^{400}, segundo a estratégia utilizada, conforme descrito a seguir.

Para a solução desta instância do problema, utilizaram-se os seguintes parâmetros:

Tabela 75: Parâmetros da reconstrução 3D (extraído de Ramos, 1992)

Parâmetro	Valor
geometria	3D
Propriedades termofísicas	α_{rad}=2.2x10^{-6} m^2/s, α_{axi}=2.2 x 10^{-7} m^2/s, k_{rad} = 6.7 W/m/K k_{axi} = 0.67 W/m/K, h=10 W/m^2/K
impulsão térmica	τ=2.0 s, q_{flux} = 15K W/m^2
modelo direto	15 x 10 x 10 nodos, Δx=0.30mm, Δy = Δz = 3.00mm, Δt=0.5s t_{total}=5 s e N = 10
modelo inverso	número de parâmetros a estimar = 400, V_{min}=5, N_{obs}=1000
constantes de inversão	u_q = 1 x 10^{-2} e l_q = 1 x 10^{-8}

Para o teste, usou-se uma disposição de defeitos em cruz, cuja representação, na forma de coeficientes térmicos de contacto é:

10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}

10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}

10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-3}	10^{-3}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-3}	10^{-3}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-3}	10^{-3}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-3}	10^{-3}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}

10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}

10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}
10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}	10^{-8}

Que pode ser representado de maneira simplificada como

8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8

8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8
8	8	8	8	3	3	8	8	8	8

8	8	8	8	3	3	8	8	8	8
8	8	3	3	3	3	3	3	8	8
8	8	3	3	3	3	3	3	8	8
8	8	8	8	3	3	8	8	8	8
8	8	8	8	3	3	8	8	8	8
8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8

8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8

8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8
8	8	8	8	8	8	8	8	8	8

Para este problema também houve a necessidade de "desnormalizar" o resultado pela inexistência dos indivíduos obtidos em Ramos (1992). Outra diferença em relação ao estudo original é que inicialmente, por restrições do pacote GALOPPS, houve que variar-se apenas os expoentes de uma base constante igual a 10.

Os parâmetros das rodadas evolutivas sâo

Tabela 76: Problema de difusão de calor, parâmetros dos testes para o caso 3D

Parâmetro	Valor
Otimização externa	busca local a cada 3 gerações, aplicada ao melhor indivíduo, com 400 tentativas positivas e 400 negativas. O incremento é 1.
Inicialização	80% dos valores iguais a 1E-8
Elitismo	1
Crossover	100% de SPAUC
Parâmetros da rodada	G=variável; P=100; PC=0.85; PM=0.03; T=15

Feitas as ressalvas acima, os resultados são

Tabela 77 : Resultados comparativos entre Ramos (1992) e um engenho evolutivo com coeficientes iguais a 1 e variando apenas os expoentes

Parâmetro	Valor em Ramos 1992	Valores do engenho evolutivo
erro logarítmico	na iteração 50, ε = 21.4429 na iteração 100, ε = 19.9233 na iteração 150, ε = 21.6004 na iteração 200, ε = 17.0381	na geração 25, ε = 19.183 na geração 50, ε = 17.220 na geração 75, ε = 16.815 na geração 100, ε = 17.098
Resíduo R(p)	na iteração 50, R(p) = 31.1 na iteração 100, R(p) = 8.98 na iteração 150, R(p) = 3.03 na iteração 200, R(p) = 2.54	na geração 25, R(p) = 8.149 na geração 50, R(p) = 6.979 na geração 75, R(p) = 6.261 na geração 100, R(p) = 6.261
tempo de CPU em segundos	na iteração 50, t =13500 seg na iteração 100, t = 27200 seg na iteração 150, t = 41000 seg na iteração 200, t = 54500 seg	na geração 25, t = 5640 seg na geração 50, t = 12060 seg na geração 75, t = 19260 seg na geração 100, t = 26460 seg

O melhor indivíduo apresentado pelo engenho evolutivo em 100 gerações, representado na forma simplificada é

1.0E-8	1.0E-6	1.0E-5	1.0E-5	1.0E-4	1.0E-5	1.0E-6	1.0E-8	1.0E-8	1.0E-8
1.0E-7	1.0E-5	1.0E-7	1.0E-4	1.0E-7	1.0E-8	1.0E-7	1.0E-6	1.0E-6	1.0E-6
1.0E-8	1.0E-8	1.0E-4	1.0E-6	1.0E-8	1.0E-7	1.0E-8	1.0E-5	1.0E-8	1.0E-8
1.0E-5	1.0E-7	1.0E-6	1.0E-6	1.0E-8	1.0E-7	1.0E-7	1.0E-8	1.0E-8	1.0E-7
1.0E-8	1.0E-7	1.0E-7	1.0E-8	1.0E-4	1.0E-6	1.0E-6	1.0E-4	1.0E-8	1.0E-6
1.0E-6	1.0E-8	1.0E-7	1.0E-7	1.0E-7	1.0E-4	1.0E-4	1.0E-7	1.0E-8	1.0E-8
1.0E-5	1.0E-8	1.0E-7	1.0E-4	1.0E-6	1.0E-8	1.0E-6	1.0E-4	1.0E-5	1.0E-5
1.0E-8	1.0E-8	1.0E-5	1.0E-6	1.0E-5	1.0E-5	1.0E-7	1.0E-4	1.0E-7	1.0E-8
1.0E-8	1.0E-5	1.0E-6	1.0E-7	1.0E-8	1.0E-5	1.0E-5	1.0E-5	1.0E-8	1.0E-7
1.0E-4	1.0E-5	1.0E-4	1.0E-6	1.0E-5	1.0E-8	1.0E-8	1.0E-8	1.0E-5	1.0E-6

1.0E-4	1.0E-6	1.0E-8	1.0E-8	1.0E-8	1.0E-5	1.0E-3	1.0E-5	1.0E-6	1.0E-7
1.0E-6	1.0E-8	1.0E-6	1.0E-4	1.0E-6	1.0E-8	1.0E-5	1.0E-6	1.0E-7	1.0E-8
1.0E-4	1.0E-6	1.0E-8	1.0E-8	1.0E-8	1.0E-5	1.0E-5	1.0E-7	1.0E-8	1.0E-8
1.0E-4	1.0E-7	1.0E-7	1.0E-3	1.0E-5	1.0E-4	1.0E-4	1.0E-5	1.0E-6	1.0E-5
1.0E-5	1.0E-7	1.0E-5	1.0E-3	1.0E-7	1.0E-3	1.0E-4	1.0E-3	1.0E-7	1.0E-4
1.0E-7	1.0E-8	1.0E-8	1.0E-5	1.0E-2	1.0E-3	1.0E-3	1.0E-4	1.0E-8	1.0E-8
1.0E-6	1.0E-8	1.0E-7	1.0E-7	1.0E-7	1.0E-3	1.0E-6	1.0E-4	1.0E-5	1.0E-8
1.0E-4	1.0E-8	1.0E-6	1.0E-8	1.0E-4	1.0E-5	1.0E-4	1.0E-8	1.0E-8	1.0E-6
1.0E-8	1.0E-6	1.0E-8	1.0E-6	1.0E-8	1.0E-6	1.0E-8	1.0E-4	1.0E-8	1.0E-8
1.0E-8	1.0E-4	1.0E-6	1.0E-5	1.0E-5	1.0E-7	1.0E-8	1.0E-7	1.0E-7	1.0E-6

1.0E-7	1.0E-6	1.0E-2	1.0E-3	1.0E-6	1.0E-4	1.0E-6	1.0E-6	1.0E-3	1.0E-6
1.0E-8	1.0E-6	1.0E-5	1.0E-7	1.0E-8	1.0E-6	1.0E-7	1.0E-7	1.0E-8	1.0E-7
1.0E-7	1.0E-8	1.0E-8	1.0E-8	1.0E-6	1.0E-5	1.0E-7	1.0E-8	1.0E-5	1.0E-5
1.0E-6	1.0E-4	1.0E-4	1.0E-6	1.0E-6	1.0E-3	1.0E-4	1.0E-4	1.0E-6	1.0E-7
1.0E-6	1.0E-7	1.0E-4	1.0E-6	1.0E-4	1.0E-2	1.0E-5	1.0E-5	1.0E-4	1.0E-8
1.0E-5	1.0E-3	1.0E-5	1.0E-8	1.0E-5	1.0E-7	1.0E-2	1.0E-4	1.0E-8	1.0E-5
1.0E-3	1.0E-8	1.0E-4	1.0E-7	1.0E-2	1.0E-6	1.0E-8	1.0E-8	1.0E-4	1.0E-6
1.0E-2	1.0E-8	1.0E-4	1.0E-5	1.0E-6	1.0E-5	1.0E-5	1.0E-7	1.0E-4	1.0E-5
1.0E-7	1.0E-4	1.0E-3	1.0E-8	1.0E-5	1.0E-4	1.0E-6	1.0E-7	1.0E-8	1.0E-6
1.0E-3	1.0E-7	1.0E-7	1.0E-7	1.0E-7	1.0E-4	1.0E-3	1.0E-2	1.0E-4	1.0E-8

1.0E-7	1.0E-8	1.0E-3	1.0E-3	1.0E-6	1.0E-2	1.0E-4	1.0E-8	1.0E-8	1.0E-8
1.0E-8	1.0E-4	1.0E-7	1.0E-3	1.0E-5	1.0E-3	1.0E-4	1.0E-6	1.0E-6	1.0E-8
1.0E-8	1.0E-4	1.0E-4	1.0E-6	1.0E-7	1.0E-7	1.0E-7	1.0E-4	1.0E-8	1.0E-7
1.0E-6	1.0E-5	1.0E-3	1.0E-5	1.0E-7	1.0E-2	1.0E-7	1.0E-3	1.0E-7	1.0E-7
1.0E-2	1.0E-7	1.0E-3	1.0E-3	1.0E-4	1.0E-2	1.0E-5	1.0E-2	1.0E-7	1.0E-7
1.0E-4	1.0E-2	1.0E-6	1.0E-4	1.0E-5	1.0E-4	1.0E-3	1.0E-7	1.0E-5	1.0E-8
1.0E-8	1.0E-5	1.0E-3	1.0E-7	1.0E-5	1.0E-7	1.0E-2	1.0E-4	1.0E-3	1.0E-2
1.0E-3	1.0E-5	1.0E-6	1.0E-8	1.0E-7	1.0E-5	1.0E-7	1.0E-8	1.0E-3	1.0E-6
1.0E-6	1.0E-5	1.0E-6	1.0E-2	1.0E-5	1.0E-5	1.0E-8	1.0E-5	1.0E-5	1.0E-8
1.0E-5	1.0E-5	1.0E-3	1.0E-4	1.0E-6	1.0E-4	1.0E-6	1.0E-2	1.0E-5	1.0E-3

Na seqüência houve a busca de representações que variassem não apenas o expoente dos coeficientes térmicos, mas sim o coeficiente por completo. A variação entre 0 e 16383 (valor máximo teórico para um alelo, do GALOPPS) revelou-se infrutífera por incapacidade do pacote, dado o tamanho deste cromossomo. A próxima etapa de teste foi uma variação de alelos entre 0-64, o que demandaria uma ocupação de 6 bits/alelo. Tampouco foi suportada pelo GALOPPS. Reduziu-se a 32 com idêntico resultado e finalmente conseguiu-se rodar o engenho com uma variação máxima de 0-16. Denominou-se a este método de mapeamento linear, e ele pode ser pensado em termos de uma reta numerada, cujos limites são u_q e l_q, tem 14 pontos no seu interior, igualmente intervalados e a distância entre cada um deles é o valor abaixo. Este fato introduz erros no resultado, pois se o intervalo de variação imposto por restrições de ordem física no problema é entre $u_q = 1 \times 10^{-2}$ e $l_q = 1 \times 10^{-8}$, uma variação de apenas 16 degraus introduzirá saltos de 0.006249994, valor muito alto, que permite apenas uma busca grosseira. Por exemplo, o valor buscado como defeito na peça é de coeficiente de $v_q = 1 \times 10^{-3}$ e usando a resolução acima, o valor mais próximo é o de 1.25×10^{-3}, o que embute, de antemão, um erro de 25%. A despeito deste fato, um resultado encontrado foi:

Tabela 78 : Resultados comparativos entre Ramos (1992) e um engenho evolutivo com mapeamento linear

Parâmetro	Valores do engenho evolutivo	Valores do engenho evolutivo
Semente	0.111	0.333
erro logarítmico	na geração 25, ε = 41.303 na geração 50, ε = 41.267 na geração 75, ε = 42.059 na geração 100, ε = 41.267	na geração 25, ε = 44.666 na geração 50, ε = 44.452 na geração 75, ε = 44.452 na geração 100, ε = 44.452
Resíduo R(p)	na geração 25, R(p) = 24.316 na geração 50, R(p) = 24.312 na geração 75, R(p) = 24.312 na geração 100, R(p) = 24.312	na geração 25, R(p) = 25.171 na geração 50, R(p) = 25.005 na geração 75, R(p) = 25.005 na geração 100, R(p) = 25.005
tempo de CPU em segundos	na geração 25, t = 5160 seg na geração 50, t = 9720 seg na geração 75, t = 19500 seg na geração 100, t = 23160 seg	na geração 25, t = 5160 seg na geração 50, t = 9720 seg na geração 75, t = 19500 seg na geração 100, t = 23160 seg

O melhor indivíduo, das duas rodadas acima, apresentado pelo engenho evolutivo em 100 gerações, representado na forma simplificada é

1.0E-08	1.0E-08	1.3E-03	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08
1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	6.3E-04	1.0E-08	1.0E-08	1.0E-08	1.0E-08
1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.3E-03	1.0E-08
1.0E-08	1.3E-03	1.0E-08	1.0E-08	6.3E-04	1.0E-08	1.0E-08	1.3E-03	1.0E-08	1.0E-08
1.3E-03	1.0E-08	1.0E-08	6.3E-04	1.0E-08	6.3E-04	1.0E-08	1.0E-08	1.0E-08	1.0E-08
1.0E-08	1.3E-03	1.0E-08	1.0E-08	1.0E-08	1.0E-08	6.3E-04	1.0E-08	1.0E-08	1.0E-08
1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08
1.3E-03	1.0E-08	1.0E-08	1.0E-08	1.3E-03	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08
1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08
1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.3E-03	1.0E-08	1.0E-08	1.0E-08	1.0E-08

1.0E-08	1.3E-03	1.3E-03	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.3E-03
1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.3E-03
1.0E-08	1.0E-08	1.3E-03	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.3E-03	1.0E-08
1.3E-03	1.3E-03	1.0E-08	1.3E-03	1.0E-08	6.3E-04	1.0E-08	1.0E-08	1.0E-08	1.3E-03
1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.3E-03	1.0E-08	1.0E-08	1.0E-08	1.0E-08
1.0E-08	1.0E-08	1.0E-08	6.3E-04	1.0E-08	6.3E-04	1.0E-08	1.0E-08	1.0E-08	1.0E-08
1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.9E-03	1.0E-08	1.0E-08	1.0E-08	1.0E-08
1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.3E-03	1.0E-08	1.0E-08
1.0E-08	1.0E-08	1.0E-08	1.3E-03	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08
1.0E-08	1.0E-08	1.0E-08	1.3E-03	1.0E-08	1.0E-08	1.3E-03	1.3E-03	1.0E-08	1.0E-08

6.3E-04	1.3E-03	1.0E-08	1.3E-03	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.3E-03
1.0E-08	1.0E-08	1.3E-03	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.3E-03	1.3E-03	1.0E-08
1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.3E-03	2.5E-03	1.0E-08	1.0E-08	1.0E-08	1.0E-08
1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08
1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.3E-03	1.3E-03	1.3E-03	1.3E-03	1.0E-08	1.0E-08
1.0E-08	1.0E-08	1.0E-08	7.5E-03	1.0E-08	1.3E-03	1.0E-08	1.3E-03	1.0E-08	1.3E-03
1.0E-08	1.0E-08	1.0E-08	1.3E-03	1.0E-08	1.3E-03	1.0E-08	1.3E-03	1.0E-08	1.0E-08
1.0E-08	1.3E-03	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08
1.3E-03	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.3E-03	6.3E-04	1.3E-03
1.0E-08	1.3E-03	1.3E-03	1.3E-03	1.0E-08	1.3E-03	1.0E-08	1.0E-08	1.0E-08	1.0E-08

1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.3E-03	1.0E-08	1.0E-08	1.3E-03	1.0E-08	1.0E-08
1.0E-08	1.3E-03	1.0E-08	1.3E-03	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08
1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.3E-03
1.0E-08	1.0E-08	1.3E-03	1.0E-08	1.0E-08	1.3E-03	1.0E-08	1.0E-08	1.0E-08	1.0E-08
1.0E-08	1.0E-08	1.0E-08	1.0E-08	6.9E-03	1.3E-03	3.8E-03	1.0E-08	1.0E-08	1.0E-08
1.0E-08	1.3E-03	1.0E-08	3.1E-03	2.5E-03	1.0E-08	1.3E-03	1.0E-08	1.3E-03	1.3E-03
1.3E-03	1.0E-08	1.0E-08	1.0E-08	4.4E-03	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08
1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.3E-03	1.3E-03	1.0E-08	1.0E-08	1.0E-08
1.0E-08	1.3E-03	1.0E-08	1.0E-08	1.3E-03	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08
1.3E-03	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.3E-03	1.0E-08	1.3E-03	1.0E-08	1.3E-03

Finalmente houve uma tentativa de aproveitar melhor a resolução máxima oferecida pelo GALOPPS. Em vez de efetuar uma variação linear, como feito acima (e que forçosamente introduziu erro pela não coincidência entre o valor dos alelos permitidos e os valores exatos buscados na reconstituição) efetuou-se um mapeamento entre os valores dos alelos e valores possíveis na busca de tal maneira a otimizar os resultados. A conversão efetuada está listada na tabela 78. Denominou-se a este teste de mapeamento discreto.

Tabela 79 - Mapeamento Discreto, conversão entre alelos e coeficientes térmicos

Alelo	Coeficiente Térmico
15	1.0E-8
14	1.0E-7
13	5.0E-7
12	1.0E-6
11	5.0E-6
10	1.0E-5
9	5.0E-5
8	1.0E-4
7	2.5E-4
6	5.0E-4
5	7.5E-4
4	1.0E-3
3	2.5E-3

2	5.0E-3
1	7.5E-3
0	1.0E-2

Obtiveram-se os resultados:

Tabela 80 : Resultados obtidos no engenho evolutivo usando o mapeamento citado na tabela 78

Parâmetro	Valores do engenho evolutivo com tamanho do torneio = 15 e 6 cortes	Valores do engenho evolutivo com tamanho do torneio = 35 e 48 cortes
erro logarítmico	na geração 25, ε = 52.1919 na geração 50, ε = 49.1426 na geração 75, ε = 45.5082 na geração 100, ε = 42.8252	na geração 250, ε = 33.4500 na geração 500, ε = 32.4800 na geração 852, ε = 31.4000
Resíduo R(p)	na geração 25, R(p) = 6.4473 na geração 50, R(p) = 5.6156 na geração 75, R(p) = 4.0738 na geração 100, R(p) = 2.8619	na geração 250, R(p) = 1.08 na geração 500, R(p) = 1.01 na geração 852, R(p) = 1.01
tempo de CPU em segundos	na geração 25, t = 5160 seg na geração 50, t = 9720 seg na geração 75, t = 19500 seg na geração 100, t = 23160 seg	na geração 250, t = 39600 seg na geração 500, t = 82800 seg na geração 852, t = 140400 seg

E o melhor indivíduo obtido nesta rodada com t=15 e 6 cortes é

1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	5.0E-07	1.0E-08	1.0E-06	1.0E-08	5.0E-07
1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-06	1.0E-08	1.0E-08	1.0E-08	5.0E-06
1.0E-08	1.0E-08	5.0E-07	1.0E-07	1.0E-08	1.0E-04	5.0E-06	1.0E-06	1.0E-08	1.0E-08
5.0E-07	1.0E-08	1.0E-08	1.0E-08	5.0E-07	2.5E-04	1.0E-08	1.0E-07	1.0E-08	1.0E-08
1.0E-08	1.0E-08	5.0E-05	1.0E-06	5.0E-05	1.0E-06	1.0E-02	1.0E-06	1.0E-08	1.0E-08
1.0E-08	1.0E-08	5.0E-05	1.0E-06	5.0E-05	5.0E-07	5.0E-07	5.0E-05	1.0E-08	1.0E-07
5.0E-05	1.0E-08	1.0E-08	1.0E-08	5.0E-06	5.0E-05	5.0E-05	1.0E-08	1.0E-08	1.0E-08
1.0E-08	1.0E-08	1.0E-08	1.0E-08	5.0E-06	5.0E-07	1.0E-06	1.0E-08	1.0E-08	1.0E-07
5.0E-07	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-07	1.0E-08	1.0E-08	1.0E-08	1.0E-08
1.0E-07	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	5.0E-07	1.0E-08	1.0E-08	1.0E-08

1.0E-08	1.0E-07	5.0E-07	1.0E-06	1.0E-08	1.0E-08	5.0E-05	1.0E-07	1.0E-07	5.0E-06
5.0E-07	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	5.0E-06	1.0E-08	1.0E-08
1.0E-08	1.0E-08	1.0E-07	1.0E-06	7.5E-04	2.5E-04	1.0E-08	1.0E-08	1.0E-08	1.0E-07
1.0E-07	1.0E-08	5.0E-07	1.0E-08	2.5E-05	5.0E-05	1.0E-08	1.0E-06	1.0E-08	1.0E-07
1.0E-07	1.0E-08	5.0E-04	7.5E-04	5.0E-04	1.0E-03	1.0E-03	1.0E-06	1.0E-08	1.0E-04
5.0E-06	1.0E-08	7.5E-04	7.5E-04	7.5E-04	1.0E-03	1.0E-03	5.0E-04	1.0E-08	5.0E-07
1.0E-08	1.0E-08	1.0E-08	1.0E-08	2.5E-05	1.0E-07	5.0E-07	7.5E-03	1.0E-08	1.0E-08
1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-05	5.0E-04	1.0E-08	1.0E-08	1.0E-08	1.0E-07
1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-08	5.0E-07	1.0E-08	1.0E-08	1.0E-08
5.0E-06	1.0E-08	1.0E-08	1.0E-08	7.5E-04	1.0E-08	1.0E-07	1.0E-08	2.5E-04	1.0E-07

5.0E-07	7.5E-04	1.0E-07	5.0E-07	5.0E-05	5.0E-06	5.0E-07	1.0E-07	1.0E-06	1.0E-06
1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-07	1.0E-06	5.0E-07	1.0E-08	1.0E-08	1.0E-08
1.0E-06	1.0E-06	5.0E-06	1.0E-06	1.0E-06	5.0E-03	1.0E-07	1.0E-08	1.0E-07	5.0E-07
1.0E-08	1.0E-07	5.0E-07	5.0E-07	1.0E-06	1.0E-06	1.0E-04	5.0E-03	1.0E-07	1.0E-08
1.0E-06	1.0E-08	1.0E-03	7.5E-04	1.0E-03	2.5E-04	2.5E-04	2.5E-03	5.0E-05	1.0E-08
1.0E-08	1.0E-08	1.0E-06	5.0E-04	5.0E-06	1.0E-04	5.0E-04	1.0E-06	1.0E-07	1.0E-07
5.0E-07	1.0E-08	1.0E-07	5.0E-07	5.0E-07	1.0E-04	5.0E-07	1.0E-08	1.0E-08	1.0E-03
1.0E-07	1.0E-03	5.0E-07	1.0E-08	1.0E-07	1.0E-06	5.0E-07	1.0E-08	1.0E-08	2.5E-04
1.0E-08	1.0E-08	1.0E-08	1.0E-08	1.0E-07	5.0E-05	1.0E-07	1.0E-08	1.0E-08	1.0E-08
5.0E-07	1.0E-07	1.0E-06	1.0E-07	1.0E-08	2.5E-04	1.0E-07	1.0E-04	1.0E-08	1.0E-07

5.0E-05	1.0E-06	1.0E-06	1.0E-08	1.0E-06	5.0E-07	1.0E-06	1.0E-07	1.0E-08	1.0E-06
1.0E-06	1.0E-07	5.0E-06	1.0E-06	1.0E-07	1.0E-08	5.0E-07	5.0E-07	1.0E-07	5.0E-07
1.0E-06	1.0E-06	5.0E-06	1.0E-07	2.5E-04	5.0E-03	1.0E-06	1.0E-07	2.5E-04	5.0E-06
1.0E-06	5.0E-06	1.0E-06	1.0E-08	1.0E-06	5.0E-06	5.0E-06	7.5E-04	1.0E-08	5.0E-07

1.0E-08	1.0E-04	1.0E-06	7.5E-04	1.0E-06	1.0E-03	1.0E-06	7.5E-04	1.0E-07	1.0E-08
1.0E-06	1.0E-07	1.0E-04	5.0E-04	5.0E-04	1.0E-07	1.0E-04	5.0E-06	1.0E-06	5.0E-07
1.0E-06	1.0E-07	1.0E-04	1.0E-06	2.5E-03	5.0E-03	1.0E-07	7.5E-04	1.0E-08	1.0E-08
5.0E-07	1.0E-08	1.0E-06	1.0E-06	5.0E-05	1.0E-06	1.0E-06	1.0E-08	5.0E-07	1.0E-06
5.0E-05	5.0E-07	5.0E-07	1.0E-06	1.0E-06	1.0E-08	7.5E-04	5.0E-07	1.0E-07	1.0E-08
1.0E-06	5.0E-06	5.0E-06	1.0E-06	1.0E-06	1.0E-08	1.0E-06	1.0E-08	1.0E-06	1.0E-08

Para o caso 3D, percebeu-se uma dificuldade representada pela deficiência de representação para o cromossomo trazida pelo GALOPPS. Este fato prejudica as conclusões obtidas, que entretanto odem ser assim listadas:

Ramos (1992) apresenta como solução para esta instância do problema um erro ε = 17.0381 e um resíduo R(p)=2.54, após 200 iterações, que demandaram uma quantidade mínima de 160.000 chamadas à função objetivo, com consumo de cerca de 15 horas de processamento.

O engenho evolutivo, pela razão acima listada, passou por 3 etapas: Na primeira variaram-se apenas os expoentes dos alelos, mantendo-se os coeficientes constantes e iguais a 1.0. Na segunda tentativa houve uma resolução de 0.006249994, valor alto pois possibilitou apenas 16 intervalos entre o maior e o menor valor. E, na terceira fêz-se um mapeamento irregular, de tal maneira a permitir que o valor buscado fosse possível de reproduzir em alguma configuração de um possível cromossomo.

No primeiro caso (só expoentes) o erro encontrado ε foi de 16.815, para um resíduo R(p) de 6.261. Este resultado foi alcançado na geração 75, embora o engenho tenha evoluído até a 100. No total foram feitas cerca de 50000 chamadas à função objetivo, demandando-se um consumo de cerca de 7h 30 min de tempo de processamento.A resolução neste caso foi de 0,01 valor alto.

No segundo caso (mapeamento linear), com resolução de 0.006249994, valor ainda alto e que não coincide com o objetivo buscado, o erro ε foi de 42,8592 para um resíduo R(p) de 24,6585. Finalmente, no último caso (mapeamento discreto), houve uma resolução variável, estabelecida *ad-hoc*. O erro ε foi de 42.8252 com um resíduo R(p) de 2.8619.

Na busca de melhoria deste resultado, conduziu-se teste com 3 alterações: aumento do tamanho do torneio de 15 para 35 (aumento da pressão seletiva), aumento no numero de cortes de SPAUC de 6 para 27 e aumento na quantidade de gerações. O resultado foi: ε = 31,40, R(p)=1.01 na geração 852, depois de 39 horas de processamento com 425.000 chamadas à função objetivo.

Como se disse acima o resultado obtido foi prejudicado pela impossibilidade de se tratar com granularidade mais fina. Isto fica evidente no resultado positivo obtido no primeiro caso em que houve significativa redução do espaço de busca pela fixação dos

coeficientes dos alelos no valor que se buscava, permitindo apenas a variação dos coeficientes, e estes em estreita faixa de busca: de -2 a -8. No segundo e terceiro casos, o resultado foi bastante pobre, sendo cerca de 3 vezes maior (para ε) na abordagem evolutiva. A comparação do resíduo quadrático também foi inconclusiva: na segunda rodada foi cerca de dez vezes maior e na terceira foi igual ao resultado obtido por Ramos (1992).

6.2.4 Conclusão para o problema de difusão de calor

Em conclusão, para este problema de difusão de calor em materiais compostos pode-se afirmar que o caso 1D é resolvido satisfatoriamente pelo engenho evolutivo. A baixa dimensão do espaço de busca (cerca de 40×10^6) certamente ajuda. Para o caso 2D, cujo espaço de busca já é maior (6.3×10^{33}) o engenho evolutivo mostra desempenho aceitável, ainda que – sob o ponto de vista do erro ε, pior. Note-se que a busca é conduzida pelo valor do resíduo, R(p) que é a soma quadrática das diferenças entre todas as temperaturas geradas pelo caso padrão e as temperaturas geradas pelo caso que se busca reconstituir. Essa é a explicação para o resultado melhor, sob o ponto de vista do resíduo R(p), do engenho evolutivo quando comparado ao obtido por Ramos (1992). A real medida de desempenho global é dada pelo erro ε que neste caso pode ser conhecida por se tratar de um caso sintético no qual a resposta correta é sabida de antemão. Mesmo nesta medida, bem como na inspeção visual o desempenho do engenho evolutivo segue sendo aceitável, ainda que pior. Para o caso 3D, houve uma degradação do modelo evolutivo. Embora os valores do resíduo R(p) sigam sendo aceitáveis quando comparados com os valores de Ramos (1992), os valores do erro ε deixam a desejar, com exceção do primeiro caso, (coeficientes constantes), cujos desempenhos em termos de ε, são praticamente iguais. Veja-se a seguir um resumo da situação.

Tabela 81 - Resultados finais para o problema 3

Caso Testado	Erro logarítmico do melhor indivíduo		Resíduo do melhor indivíduo	
	Ramos (1992)	*Engenho evolutivo*	Ramos (1992)	*Engenho evolutivo*
Caso 1D	0.0	*0.0*	0.0	*0.0*
Caso 2D	4.260	*9.479*	0.706	*0.524*
Caso 3D	17.038	----------	2.540	----------
Caso 3D - variação de expoentes	----------	*16.815*	----------	*6.261*

Caso 3D - mapeamento linear	----------	*41.267*	----------	*24.312*
Caso 3D - mapeamento discreto	----------	*42.825*	----------	*2.861*
Caso 3D - t=35 e 27 cortes	----------	*31.400*	----------	*1.010*

7. Idéias para uma Metodologia

7.1 Verificação de convergência

Uma proposta inicial é obter uma solução completa para o problema que se está estudando. Não importa como ela foi obtida, se é um caso real ou fictício, se ocorre ou não nas instâncias do problema que estão sendo estudadas. Com este resultado em mãos, e com o engenho evolutivo a postos, começam as tratativas de solução.

Pode-se começar incluindo na população inicial da evolução uma quantidade não nula de indivíduos que já são a solução do problema. Se o engenho evolutivo estiver funcionando bem, esta solução, além de não se perder, deve fecundar progressivamente os demais componentes da população de maneira que, em um determinado número de gerações, haja a convergência em direção ao resultado correto.

Um teste interessante que pode ser realizado é variar a quantidade percentual de indivíduos-solução dispersos entre uma população aleatoriamente gerada. Na medida em que este número cresce, deve diminuir a quantidade de gerações necessária para obter a convergência final. Em resumo este ataque propõe que uma determinada percentagem dos indivíduos da população esteja 100% correta.

7.2 Criação de convergência

Em abordagem similar ao caso anterior, mas ligeiramente diferente, tem-se uma população inicial inteiramente aleatória. Nesta, e novamente por diversos mecanismos possíveis, vai sendo introduzida uma tendência mais ou menos forte em direção ao resultado final. Agora, em momentos aleatórios (mas diferentes do momento zero), certos alelos vão sendo transformados no valor final que deverão ter ao final da rodada.

A finalidade principal desta abordagem é o estabelecimento prévio de convergência. Precisa-se desta situação para escolher quais parâmetros básicos usar no início da pesquisa. Mais tarde, é possível e até provável que estes números sejam modificados, mas não importa: a convergência compulsória determinada pelo esquema acima sugerido, se presta à escolha de taxas (tipicamente de crossover, mutação) e de seus parâmetros associados, assim como métodos de seleção, de crossover e de mutação.

Se não se fizer algo parecido com o aqui descrito pode-se estar em uma situação de *deadlock* do tipo: Não se pode testar comparativamente os diversos operadores por-

que não há convergência, e não há convergência porque não se dispõe de operadores adequados para utilizar.

A maneira que foi usada aqui para obter a convergência compulsória foi a modificação da rotina de mutação. Nesta nova rotina, introduziu-se um parâmetro que determina a probabilidade de se usar a mutação antiga (a convencional) ou a nova (a que sempre inclui no alelo em que está atuando o valor esperado). Assim quando este parâmetro era 1.0 não havia modificação da mutação e quando ele era 0.0 toda vez que ocorresse uma mutação seria gerado o valor correto.

Finalmente, mas bem importante ainda, esta abordagem permite testar, isolar e corrigir erros de programa, que são tradicionalmente difíceis de tratar em programas que implementam algoritmos genéticos. Esta dificuldade é determinada por no mínimo 2 fatores: a complexidade dos algoritmos e do código, associado ao caráter estocástico dos resultados.

7.3 Hibridização

Se for possível evitar esta abordagem, tanto melhor, pois um grande volume de trabalho não será necessário. Lastimavelmente, parece que apenas problemas simples, ou instâncias simples de problemas médios ou complexos podem ser tratados através de engenhos evolutivos a que se pode chamar "puros" (sem hibridização). Em se fazendo necessária, a hibridização pressupõe um conhecimento adequado do problema a resolver e a criação de uma ou mais das pré-condições da CE, quais sejam:

- Gerador de soluções que "saiba" como resolver o problema e que gere porções horizontais ou verticais da população que sejam respostas adequadas ao problema
- Esquema de codificação que facilite ou induza a obtenção de soluções
- Operadores evolutivos adicionais ou modificação dos já existentes de maneira a forçar o surgimento de soluções

É importante que todo novo melhoramento introduzido esteja associado a uma taxa probabilística. Assim, quando utilizado com a taxa = 1.0, todas as oportunidades possíveis serão atendidas pelo melhoramento hibridizado. Quando a taxa for 0.0, em nenhuma oportunidade o código novo será chamado. Agindo assim, preserva-se um modelo evolutivo "puro" associado a eventuais hibridizações, variando-se a escala entre estes dois extremos através de uma simples taxa.

8. Conclusão

8.1 Quanto aos problemas estudados

8.1.1 Robustez em Relação aos Parâmetros de Rodada

A primeira conclusão consistente com praticamente todos os testes efetuados é a relativa independência da convergência do engenho evolutivo sobre os parâmetros exigidos pela computação evolutiva. Dito de outra maneira, pôde-se verificar a robustez do método evolutivo em relação aos diferentes valores atribuídos aos parâmetros que governam a busca evolutiva. Depois de implementados os três operadores aqui descritos, dado um conjunto de parâmetros que se poderia chamar razoável -- por exemplo, seguindo as recomendações da literatura -- o algoritmo converge e segue convergindo a despeito de pequenas alterações não estruturadas efetuadas sobre estes valores.

Durante o desenvolvimento do trabalho, diversas estratégias foram sendo desenvolvidas com vistas à otimização do resultado. Assim, por exemplo, para o problema 1, foram criadas a busca local com garantia de melhora (chamada V2 no desenvolvimento do texto) e a mutação variando de acordo com a profundidade. Para o problema 3, utilizaram-se diversas modalidades de codificação, com vistas a compatibilizar restrições do recursos usado (o GALOPPS) com exigências do problema em estudo. Todas estas iniciativas tiveram resultados conclusivamente melhores, porém pequenos. Entretanto, coerente com o título deste ítem (robustez com relação aos parâmetros), nenhuma destas abordagens trouxe resultados muito significativos, pelo que os métodos originalmente estabelecidos foram preservados.

8.1.2 Desempenho dos Operadores Especialmente Desenvolvidos

Quanto aos operadores especialmente desenvolvidos, e iniciando por SPAUC (Spatial Uniform Crossover), pode-se afirmar que quando comparado com a recombinação Uniforme, no âmbito do problema geofísico, aquele tem desempenho melhor que este, tanto na média, quanto no melhor caso. Esta conclusão permanece válida haja ou não ruído. SPAUC funciona melhor ao permitir que partes homogêneas de ancestrais sejam trocadas em bloco. Tudo se dá como se os *building blocks* citados por Goldberg (1989) tivessem 2 ou 3 dimensões, este último caso para o problema 3.

Para os problemas 1 e 2, não houve uma busca exaustiva na quantidade ideal de cortes para o funcionamento de SPAUC, fazendo-se apenas ajustes em torno aos

valores originalmente estabelecidos de 2 cortes horizontais e 3 verticais. Para o problema 3, no caso 2D, manteve-se a quantidade de cortes em 2 x 3. No caso 3D, iniciou-se com este mesmo padrão de cortes e diante do insucesso, passou-se a utilizar uma quantidade de cortes que fosse proporcional ao tamanho do indivíduo. Assim, se para indivíduos de 70 gens, (problema 1) usaram-se 6 cortes, para indivíduos de 400 gens, (caso 3D do problema 3) utilizou-se 27 cortes, distribuídos nos 3 planos, ou 3 x 3 x 3 cortes.

Quando se passou do problema 1 para o 2 e principalmente para o 3, houve a preocupação de manter inalterados todos os parâmetros, algoritmos, conceitos, operadores e técnicas utilizadas. Agiu-se assim para permitir comparação de resultados e principalmente a generalização da técnica.

Quanto à homogeneização, os testes conduzidos permitiram concluir pela sua eficácia, naturalmente, nos casos em que a solução buscada é homogênea. Notou-se que, para estes casos, quanto maior a quantidade de ciclos de homogeneização (aplicados ao melhor indivíduo), ou quanto maior a probabilidade de se efetuar operações de homogeneização sobre os descendentes resultantes de SPAUC, melhor é o resultado. A contrapartida é que esta operação consome recursos significativos ao exigir chamadas sucessivas à função objetivo. A solução de equilíbrio está em um meio termo entre estes dois compromissos.

Finalmente, quanto ao terceiro operador especialmente produzido, a busca local, deve-se ressaltar também a sua importância ao permitir melhorias constantes e firmes do melhor indivíduo da população. Por característica da computação evolutiva, (recombinação) logo após estas melhorias terem sido introduzidas pela busca local no melhor indivíduo elas se distribuiam pelos demais membros da população. Assim, se a busca local introduz uma melhora determinística aplicada a um único indivíduo, as recombinações posteriores, introduzem melhorias estocásticas mas agora aplicadas potencialmente a toda a população.

8.1.3 Robustez em Relação à Inicialização

Quanto à inicialização dos valores da população evolutiva, percebe-se que na busca do erro diretamente mensurável pelo engenho evolutivo (resíduo) é ajudada pela inicialização com valores próximos ao que se busca reconstituir. Esta ajuda está presente, na computação evolutiva, ainda que tenha menor importância do que para os algoritmos " clássicos". Nestes, o início em pontos muito afastados do que se busca reconstituir simplesmente impede o algoritmo de encontrar uma solução. Sob este aspecto

o engenho evolutivo é mais robusto. Finalmente, quando se considera o erro indireto (a diferença da solução encontrada e daquela que era procurada), os melhores resultados também são obtidos quando a maioria da população é inicializada com valores próximos à solução buscada, se bem que um mínimo de aleatoriedade nos dados de parte da população ajudam sobremaneira no sucesso da busca.

8.1.4 Robustez em Relação ao Ruído

Quanto à imunidade ao ruído, se se analisar o desempenho de um conjunto de testes, iniciando com valores pequenos de ruído e prosseguindo com o incremento destes valores, percebe-se no âmbito do problema 1 que para baixos valores de ruído há uma equivalência nos valores médios de desempenho de ambos os algoritmos. (No melhor desempenho de cada rodada o algoritmo clássico é melhor). Para valores substancialmente maiores de ruído, tanto o desempenho médio quanto o melhor resultado são os do engenho evolutivo. No problema 3, este aspecto não pôde ser considerado em sua plenitude, pois quando o ruído esteve sob comparação (no caso 1D) o desempenho de ambos os métodos foi similar. Para os casos 2D e 3D não houve comparações variando-se o ruído.

8.1.5 Desempenho em soluções Irregulares

Quanto à irregularidade da solução buscada, notadamente no problema 1 e neste, para o caso 3, aquele que busca reconstituir um indivíduo altamente irregular na sua composição, o engenho evolutivo mostra sua superioridade: em praticamente todas as rodadas, ele teve desempenho superior quando comparado ao algoritmo clássico.

8.1.6 Critério de Parada

Quanto ao critério de parada do engenho evolutivo, usou-se inicialmente aquele sugerido pela literatura que é a estabilização do erro que se busca minimizar. Entretanto posteriormente conduziu-se estudo analisando o erro indireto e pôde-se perceber que este se estabilizava antes do que o resíduo. Assim, por exemplo, embora todas as rodadas do problema 1 tenham tido 150 gerações, verificou-se que para este problema poderiam ter sido apenas 50. Esta conclusão não pode ser generalizada, já que este ponto não foi estudado. Deve-se lembrar também que esta conclusão só é possível quando se conhece de antemão a solução que se busca, o que nem sempre ocorre. A recomendação, portanto, é que critérios adequados de parada sejam estabelecidos para cada problema em particular. Idealmente, e esta questão fica como sugestão para o prosseguí-

mento deste estudo, poder-se-ia buscar alguma heurística, baseada talvez no nível de ruído empregado que informe quando a busca poderia ser encerrada.

8.1.7 Eficiência Computacional

Quanto à eficiência computacional, para o problema 1, o estudo inicialmente apontou um consumo global de recursos (abstraindo-se o tempo eventualmente gasto na programação e depuração do modelo) na ordem de 5 a 6 vezes em média, maior do engenho evolutivo quando comparado a algoritmos não evolutivos para o mesmo problema. Considerações posteriores reduziram para 1/3 o consumo demandado pelo algoritmo evolutivo, representando afinal, um consumo médio maior do algoritmo evolutivo sobre o clássico. Este valor se justifica pelo fato do engenho evolutivo trabalhar sobre uma população de soluções e não apenas sobre uma única solução candidata. A medida usada para esta comparação é a da quantidade de chamadas à função objetivo, que representa -- de longe -- o maior recurso consumido. Não se efetuaram comparações no tempo gasto pelo modelo (tanto evolutivo quanto clássico) por esta medida ser de difícil comparação e de não ser relevante frente à quantidade de invocações da função objetivo. Para os problemas 2 e 3 esta questão não se colocou.

8.1.8 Tamanho dos Problemas

Os testes conduzidos, principalmente para os problemas 2 e 3, sugerem que à medida em que o tamanho do problema cresce, aumenta a dificuldade do engenho evolutivo em encontrar possíveis soluções. Esta tendência se encontra de acordo com o "senso comum" que sugere que quanto maior a dimensão do espaço de busca, mais especializado e/ou robusto tem que ser o método de busca e solução do problema. Não houve como comparar tamanhos para o problema 1 (o mais estudado), pois não foi possível executar o modelo clássico para instâncias maiores deste problema.

8.1.9 Avaliação do Uso do GALOPPS

Quanto ao produto usado neste estudo, o GALOPPS, inicia-se dizendo ter sido uma decisão original de projeto a utilização de um pacote pré-existente e de uso já consagrado. Embora -- na fase de estudo inicial -- tivesse havido o desenvolvimento de um

programa evolutivo, que foi nomeado de TUDEL[7], este foi construído apenas com finalidade didática, para consolidar e explorar os conceitos do campo.

No momento da programação dos modelos, havia que ter confiança total no *software* e para obter tal confiança no TUDEL, um programa rigoroso de testes e depuração teria que ser implementado: não teria havido tempo hábil. Assim, usou-se o GALOPPS. O produto é confiável. Tem código bem (auto) documentado, embora o mesmo não se possa dizer da sua documentação em papel (manual).

Como todo software, ele é um compromisso entre generalidade e desempenho. Funcionou e bem para tamanhos pequenos e médios dos problemas. Indivíduos com 70 valores (problema 1), 81 valores (problema 2) e 40 valores (problema 3) funcionaram bem e com precisões aceitáveis. Instâncias maiores (400 valores, no caso 3 do problema 3) não tiveram o mesmo sucesso, houve que se fazer concessões que prejudicaram o resultado obtido. Para fazer frente a este último caso, ou mais genericamente, a problemas muito grandes, haveria que reprogramar o pacote (possível, pois o GALOPPS é distribuído na modalidade fonte), mas correndo o risco de efetuar alterações profundas no seio do programa.

Quanto ao tipo de armazenamento interno, há que se separar os programas que implementam computação evolutiva em 3 classes:

- armazenamento apenas binário, com alfabeto de cardinalidade = 2, ficando a responsabilidade de agrupar e interpretar os gens por conta do usuário
- armazenamento inteiro com alfabeto de cardinalidade maior que 2, ficando a responsabilidade da conversão de valores inteiros para reais por conta do usuário
- armazenamento dos valores diretamente em formal real ou de ponto flutuante.

O GALOPPS atende apenas aos dois primeiros critérios e nesta tese o segundo sempre foi o utilizado. Desde que a cardinalidade do alfabeto seja ampla o suficiente para atender a todos os alelos possíveis na precisão desejada, a segunda alternativa pode ser adequada, desde que se ignore o pequeno *overhead* representado pelas conversões de inteiro para real.

8.1.10 Para Encerrar

Em resumo e para concluir, pode-se afirmar, ainda que pagando algum preço de generalização ao se obter a concisão a seguir:

---

[7] acrônimo obtido com o nome dos dois cientistas considerados seminais nesse contexto: Turing e Mendel.

Quanto menos regular for o indivíduo buscado, quanto mais ruidoso for o ambiente sob estudo, quanto mais incerta for a inicialização da população (ou da solução) inicial, mais adequado é o enfoque evolutivo, que pode portanto ser caracterizado como robusto à condição inicial, robusto ao ruído e robusto à irregularidade da solução.

8.2 Genericamente quanto à computação evolutiva

Seguem-se algumas conclusões genéricas, sobre a computação evolutiva, obtidas de maneira paralela ao estudarem-se os problemas acima citados.

8.2.1 A CE Funciona

A primeira conclusão, é que a despeito das dificuldades de sua implementação, a computação evolutiva funciona. Similarmente à evolução natural, para a qual não se conhece o ritmo, o objetivo ou a finalidade, também aqui necessariamente não se sabe por quais caminhos o algoritmo chegou a uma solução. Conhece-se o processo em tese, mas para uma dada execução (ou um conjunto de execuções) apenas o minucioso acompanhamento *a posteriori* é que poderá dizer algo sobre o que ocorreu. Independente de seus méritos intrínsicos a computação evolutiva, ao se relacionar com a evolução natural e ao permitir manipulações numéricas baseadas na evolução enriquece ambas as ciências. Com efeito, pontes sobre temas do conhecimento humano originalmente independentes, têm o seu valor, pelo que de luz jogam sobre ambos os campos. Num contexto similar, M. Minsky um dos pais da Inteligência Artificial disse "*A prova da equivalência de duas ou mais definições têm sempre um efeito vinculativo quando as definições provêm de experiências e motivações diferentes*" (1967).

A computação evolutiva a despeito de obedecer ao princípio de Lanczos (1961), ou ao equivalente mais moderno, o "*No-free-lunch theorem*" de Wolper e Macready, citado em (Schwefel, 1997), pela qual o programa está *a priori* impedido de fornecer respostas corretas quando faltem dados necessários à geração de tais respostas, oferece um equilíbrio aceitável entre investimento intelectual na preparação dos modelos e na qualidade do resultado devolvido.

Finalmente, a grande qualidade da computação evolutiva é a sua vocação pelo domínio do complexo, do não previsível, do robusto. Parafraseando Hofstadter, "*Todos os caminhos estão abertos para a computação evolutiva*" (1995).

8.2.2 Operadores *ad-hoc* são Necessários

Ainda que seja tentador usar um algoritmo canônico (representação binária, operadores de seleção, recombinação e mutação convencionais) que já se encontra pronto para uso (como em Goodman, 1996) o qual pode gerar resultados corretos em curto espaço de tempo, a prática demonstrou que esta abordagem só é apropriada para problemas simples. Para problemas mais complexos -- em particular para problemas inversos com alta dimensionalidade -- resultados satisfatórios somente foram obtidos quando conhecimento específico do problema foi introduzido no engenho evolutivo.

Aqui se está diante de um dilema, pois quanto mais conhecimento específico do problema for introduzido no algoritmo evolutivo menos genérico e possivelmente menos robusto, este passa a ser. A generalidade é um grande objetivo ao se programar e a robustez parece ser uma das melhores qualidades da computação evolutiva. Desse dilema se sai usando uma estratégia a que se poderia chamar de "conhecimento em gotas". Trata-se de ir agregando pequenas porções de conhecimento e somente até que o programa chegue a resultados aceitáveis.

Esta estratégia foi aqui seguida: usou-se o mínimo indispensável de informações sobre o entorno do problema. Os 3 operadores que introduzem conhecimento do problema no engenho evolutivo (homogeneização, busca local e SPAUC) foram necessários. O primeiro introduz algum grau de regularização nas soluções, o segundo permite refinar bons candidatos à solução e o terceiro permite e troca de blocos construtivos (*building blocks*) regulares entre os participantes de um *crossover*.

Para estes operadores as únicas peças de conhecimento empregadas foram:

- Os indivíduos são n-dimensionais e devem ser manuseados como tal
- Distribuições de condutividade sobre a crosta terrestre seguem padrão homogêneo
- Algoritmos simples de subida-da-encosta funcionam e ajudam a melhorar resultados

Contrasta a primeira característica com Bäumer (1996), no qual a computação evolutiva foi usada no problema de inversão magnetotelúrica mas com abordagem mais limitada, apenas uni-dimensional.

8.2.3 CE é Robusta

Para este estudo rodaram-se mais de 200 vezes um conjunto completo de execuções evolutivas envolvendo os 3 problemas, mas principalmente o primeiro. Na grande maioria das vezes, a despeito dos valores mais díspares que tenham sido fornecidos

como parâmetros ou ainda a despeito de pequenos erros existentes no código, ainda assim, na maioria das vezes o engenho convergiu em direção ao valor correto. Certamente bem poucas abordagens da ciência da computação podem apresentar comportamento similar.

8.2.4 Resultados são Competitivos

Quando se comparam os resultados entre a abordagem clássica e a evolutiva, verifica-se que esta segunda é mais robusta quanto a imunidade ao ruído bem como as condições iniciais da solução (caso clássico) ou da população de candidatos (evolutivo), entre outros critérios. De início a abordagem clássica tem como vantagem inconteste o menor consumo de recursos computacionais. Não poderia ser diferente: algoritmos clássicos trabalham sobre uma solução, enquanto os evolutivos o fazem sobre uma população.

Esta "deficiência" ao final revela-se a maior virtude do método. Pois muitas soluções garantem a diversidade e o paralelismo na busca. E, mesmo esta sobrecarga de trabalho, conforme mostrado aqui, não se revela excessiva. O dobro ou o triplo do consumo de computação parece ser um patamar aceitável e este consumo pode vir a ser a chave de acesso a resultados mais precisos e portanto mais valiosos.

Seja como for, para os casos estudados aqui, que lidam com dados sintéticos, ambas abordagens têm seu mérito. Entretanto em dados do mundo real nos quais a precisão venha a se tornar uma questão chave, a técnica evolutiva aqui apresentada certamente será uma alternativa interessante. Outra possibilidade, sugerida por Bäumer (1996), é utilizar a computação evolutiva para localizar uma região promissora no espaço de busca e então executar uma busca local eficiente para localizar o melhor ponto nessa região do espaço de busca.

8.3 Direções a seguir

Para a continuação deste trabalho levantam-se algumas possibilidades, a seguir categorizadas. Antes de ir a elas, deve-se citar uma continuação imperativa: a melhora dos resultados do problema de difusão de calor em materiais compostos, uma vez que os encontrados não atingiram o mesmo nível qualitativo daqueles obtidos no problema das condutividades geoelétricas. Acredita-se ser plenamente possível melhorá-los desde que os processos direto e inverso sejam estudados em maior profundidade. Esta obser-

vação encontra respaldo no fato de que o problema 1 foi estudado por cerca de 2,5 anos, enquanto o problema 3 ficou em análise apenas cerca de 6 meses.

8.3.1 Estabelecimento de Critérios de Parada

Considerando que algoritmos evolutivos são consumidores de recursos de computação, passa a ser de vital importância estudar, propor e validar métricas associadas a cada problema estudado que sugiram critérios de parada efetivos. Como se viu acima, o uso de critérios genéricos e consagrados pode levar a um sobreconsumo de recursos. Para grandes espaços de busca, este sobreconsumo pode se tornar proibitivo e impedir o uso da ferramenta evolutiva. Alternativamente, pode-se buscar um método -- ainda que impreciso -- que permita predizer condições genéricas de parada, a partir da análise estatística da população, do ruído envolvido e eventualmente de outros parâmetros dos problemas em estudo.

8.3.2 Problemas com Alta Multimodalidade

Certa classe de problemas, aqui representadas pelo problema 2, ainda demandam melhores tratamentos computacionais para poderem ser abordadas usando CE, como por exemplo abordado em Schwefel (1997a). O problema 2 tinha duas ou mais configurações iniciais completamente diferentes gerando o mesmo valor da função objetivo, o que *a priori* o caracteriza como altamente multimodal, e traz dificuldades para sua abordagem sob a computação evolutiva "convencional". A favor da CE, diga-se entretanto, que a solução desta classe de problemas é muito difícil, seja qual for o método empregado. Novos operadores (que tratem apropriadamente a multimodalidade) e funções objetivo mais robustas podem ser propostas de continuação desta tese.

8.3.3 Mais Problemas com Indivíduos Bi e Tridimensionais

Outros problemas inversos em cuja representação das soluções as relações de vizinhança sejam importantes precisam ser estudados usando o ferramental aqui proposto. Especificamente falando, o relativo insucesso do problema 3, deveria ser tratado. Pode-se especular quanto aos motivos do ocorrido: Ou uma inadequação do problema em estudo aos operadores n-dimensionais (e neste caso, tridimensionais), ou talvez uma dificuldade de SPAUC de tratar o problema em questão. Em Oliveira (1996), há uma lista de problemas correntemente sendo estudados, alguns dos quais podem ser manipulados visando sua solução usando o ferramental aqui proposto.

8.3.4 Outros Operadores Evolutivos

Uma conclusão do trabalho aqui desenvolvido diz respeito à robustez do método evolutivo quando usado junto com heurísticas e/ou representações específicas do problema sob tratamento. Como disse Davis (1991), um algoritmo evolutivo híbrido é melhor que um algoritmo evolutivo padrão e também pode ajudar um algoritmo específico para resolver o problema em estudo, desde que este sozinho seja incapaz de achar uma solução, seja pelo tamanho do problema, pela presença de ruído ou por qualquer outro motivo. Como se discutiu no texto, a tentativa aqui sempre foi a de usar o mínimo de conhecimento do problema de maneira a ter soluções tanto quanto possível, genéricas. Entretanto, nada impede que se use estratégia oposta, agregando muito conhecimento do problema ao seu solucionador e usando a CE apenas no que ela tem de essencial (paralelismo, processos estocásticos e pressão evolutiva variável, apenas para ficar em algumas características, por exemplo). Esta proposta, quiçá para os mesmos problemas aqui estudados, representaria um excelente termo de comparação entre as 3 alternativas: métodos clássicos, métodos evolutivos com pouco conhecimento agregado e métodos evolutivos fortemente especializados.

8.4 Contribuições

Inicialmente, esta tese reforça o tratamento de problemas inversos através da computação evolutiva. Embora ambos sejam temas importantes e que têm sido tratados de maneiras diversas, a bibliografia revela não muitas iniciativas agregando os dois campos do conhecimento. Mais especificamente, o problema intitulado inversão magnetotelúrica, embora tenha sido objeto de muito estudo e investimento, fundamentalmente graças à importância econômica de sua solução, ainda não havia sido resolvido como o foi aqui, através da consideração de características geométricas bidimensionais das distribuições de condutividade no subsolo. Paralelamente, a apresentação de resultados evolutivos lado a lado com resultados de técnicas clássicas, permitindo a comparação de resultados e desempenhos, não foi encontrada em nenhuma fonte da pesquisa bibliográfica. Esta constatação permite concluir que se não inédita, esta apresentação de resultados, pelo menos, é rara na literatura.

Também, e agora sob o prisma da computação evolutiva, apresentam-se os dois operadores evolutivos especialmente desenhados para a solução do problema geofísico. Tratam-se dos operadores SPAUC e de homogeneização. O primeiro, ao permitir uma generalização suave do operador de *crossover* uniforme e o segundo, ao agregar ao processo de otimização local uma melhoria graças à existência de padrões homogê-

neos na solução buscada permitiram melhorar a solução anteriormente obtida em um engenho evolutivo convencional de maneira tal a que se pudesse considerar o problema como resolvido, coisa que anteriormente não acontecia. Considerando que ambos são utilizados juntos na operação da recombinação e considerando que esta é uma operação fundamental em se tratando de algoritmos evolutivos, a apresentação de SPAUC e da recombinação em termos conceituais, a discussão de seus parâmetros internos e a ilustração com resultados comparativos pode ser considerada a principal contribuição desta tese.

8.5 A tese em retrospecto

Esta tese se iniciou um janeiro de 1996 e se encerrou em fevereiro de 2000, demandando portanto pouco mais de 4 anos de trabalho. O texto de qualificação, apresentado à UFSC em 30 de agosto de 1996, teve 17 versões, enquanto a tese teve 15.

A cada modificação importante do código ou dos resultados encontrados, uma nova estrutura de diretórios era criada. Os resultados finais apresentados encontram-se no 13º diretório criado.

O trabalho se iniciou com um computador equipado com processador INTEL 386, com ciclo de *clock* de 25 MHz e equipado com um disco de 0.3 GBytes. Após inúmeras substituições e ampliações chega-se ao final com um equipado com processador Pentium III, com disco de 8.4Gbytes e ciclo de 500Mhz.[8] Totalizando todas as execuções, e considerando os seus tempos médios, estima-se ter gasto cerca de 4500 horas de processamento em uma máquina de 100 MHz, para se ficar na média de potência.

Utilizaram-se adicionalmente computadores da Companhia de Informática do Paraná, quando os volumes de processamento paralelo inviabilizaram a utilização de um único computador. Usou-se também o computador SUN *Sparc*, denominado ALIEN localizado no Instituto Nacional de Pesquisas Espaciais em São José dos Campos, de modo remoto, através das ferramentas TELNET e FTP.

As linguagens empregadas na programação do modelo foram o C++ e o FORTRAN. Aquele foi o Borland C++ versão 3.2 enquanto este foi o Microsoft Fortran Workbeench, versão 1.0. Adicionalmente os textos foram produzidos em um editor de textos Microsoft WORD, versão para Windows 95. Um processamento de apoio importante foi efetuado utilizando-se a linguagem APL, versão 6.4. Para os gráficos usou-se o software GNUPLOT, versão 3.4. Alguns desenhos foram feitos usando-se ABC

[8] Paradoxalmente, este último equipamento teve custo inferior aquele.

Flowcharter, versão 2.0. Textos foram produzidos para congressos internacionais usando-se POSTSCRIPT e para sua visualização usaram-se os *softwares* Aladin Ghostscript versão 3.4 e GSView, versão 2.1

O diretório de resultados apenas do problema 1, contém 2031 arquivos. Tipicamente cada 8 arquivos documentam uma rodada, pelo que, há cerca de 250 execuções guardadas. Os arquivos mostram a coleção de parâmetros iniciais, o desenvolvimento da busca evolutiva e o resultado alcançado.

O conjunto total de diretórios e arquivos envolvidos com o curso de doutorado alcança o volume de 117 Mbytes, sem considerar as diversas cópias de segurança existentes.

Morando em Florianópolis e depois em Curitiba, e trabalhando em conjunto com o grupo PINGA (Problemas Inversos e Algoritmos Genéticos, no âmbito da UNIVAP e INPE) em São José dos Campos, o deslocamento geográfico demandou cerca de 65 viagens Curitiba - Florianópolis e 27 Curitiba - São José dos Campos, totalizando a marca de 72.000 Km percorridos.

Este relato só pode ser encerrado afirmando que para iniciativas complexas e demoradas como um curso de doutorado sempre se aplica a Lei de Hofstadter que diz textualmente: *"isto sempre toma mais tempo do que se espera, a despeito de que se leve em consideração a Lei de Hofstadter"* [9].

---

[9] dito em *Göedel, Escher e Bach*, livro de Douglas J. Hofstadter.

9. Anexos

9.1 Anexo A - código fonte de SPAUC

```
#include "external.h"
char which_crossover[] = "spauc.c";
void crossover(parent1, parent2, child1, child2, jcross)
unsigned *parent1, *parent2, *child1, *child2;
int jcross[2];
{
 int counter1, counter2, bitsleft, UintSize, fieldnum, lclfldlength;
 int prevfldlength;
 int rememberflip;
 unsigned mask;
 double PROBSPAUC;
 // PROBSPAUC e a probabilidade de SPAUC ser chamada. Se = 0, funciona
 // como o Unifx. Se 0.3, em 30% dos casos SPAUC eh chamada e em 70% a chamada
 // e a UNIFX. Se = 1.0, em 100% dos casos SPAUC e' chamada
   PROBSPAUC = 1.0;
 //           #####
 if (flip(PROBSPAUC)) {
   SPAUC(parent1, parent2, child1, child2, jcross);
 }
 else {
  ... segue o código de "uniform crossover convencional"
 }

void spauc(parent1, parent2, child1, child2, jcross)
unsigned *parent1, *parent2, *child1, *child2;
int jcross[2];
{
 int LPP, CPP, LPX, CPX, ILP, ICP, DISTRIBUI,
    I, J, K, LIN, COL, XX, IMPRIME, ISA, ij, ik;
 int LP[6];
 int LX[8];
 int CP[9];
 int CX[11];
 int DISTA [7] [10];
 int VETBOL[70];
 int VETSAI[140];
 int jk;
 int intsp[70];
 int counter1, counter2, bitsleft, UintSize, fieldnum, lclfldlength;
 int prevfldlength;
 int rememberflip;
 unsigned mask;
// LPP = quantidade de linhas. Numero inteiro entre 1 e 6 inclusive
   LPP = 2;
// CPP = quantidade de colunas. Inteiro entre 1 e 9, inclusive
   CPP = 3;
   randomize();
   LP[0] = 1 + rnd(0,(5-LPP));
   for (I=1;I<LPP;I++) {
     LP[I] = LP[I-1] + 1 + rnd(0,(7 + I - (LPP+LP[I-1])));
   }
   randomize();
   CP[0] = 1 + rnd(0,(7-CPP));
   for (I=1;I<CPP;I++) {
     CP[I] = CP[I-1] + 1 + rnd(0,(10 + I - (CPP+CP[I-1])));
   }
   DISTRIBUI = 0;
   ICP = 0;
   ILP = 0;
   I = 0;
   while (I < 7) {
      J = 0;
```

```
      while (J < 10) {
            if (J == CP[ICP]) {
               DISTRIBUI++;
               ICP++;
            }
            DISTA [I] [J] = DISTRIBUI;
            J++;
      }
      I++;
      while ((I != LP[ILP]) && (I < 7)) {
          for (K=0; K<10; K++) {
             DISTA[I] [K] = DISTA [I-1] [K];
          }
          I++;
      }
      DISTRIBUI++;
      ILP++;
      ICP=0;
    }
    for (I=0;I<DISTRIBUI;I++) {
      VETBOL[I] = flip(0.5);    // seria o flip()
    }
    J = 0;
    I = 0;
    LIN = 0;
    COL = 0;
    while (I<70) {
      if (COL > 9) {
          COL = 0;
          LIN++;
      }
      XX = DISTA[LIN] [COL];
      VETSAI[I] = VETBOL[XX];
      I++;
      COL++;
    }
    ISA = 0;
```

... aqui segue o código normal de uniform crossover que é executado pela SPAUC. A diferença está em que na função crossover original, em determinado momento há uma chamada à função flip(0.5) que visa gerar um binário aleatório.

Agora esse ponto foi modificado de modo a em vez de chamar flip(0.5) consultar uma matriz (de nome VETSAI), especialmente preparada pela SPAUC.
...

```
}

// alteracao a 12/09/98
LPX = acertaLP(LPP,LP,LX);
CPX = acertaCP(CPP,CP,CX);
chromtointarray(intsp,child1);
    LPX--;
    CPX--;
    for (ij=0;ij<LPX;ij++)
      {
      for (ik=0;ik<CPX;ik++)
        {
          if (flip(0.05))  // PROBABILIDADE DE MANIPULAR ESTE BLOCO
//                    #####
            {
             regula((CX[ik]+1),CX[ik+1],(LX[ij]+1),LX[ij+1],intsp);
            }
        }
      }
    intarraytochrom(intsp,child1);
    chromtointarray(intsp,child2);
    for (ij=0;ij<LPX;ij++)
      {
      for (ik=0;ik<CPX;ik++)
        {
```

```
          if (flip(0.05)) // PRABILIDADE DE MANIPULAR ESTE BLOCO
//                    ####
              {
               regula((CX[ik]+1),CX[ik+1],(LX[ij]+1),LX[ij+1],intsp);
              }
         }
       }
    intarraytochrom(intsp,child2);
}


int acertaLP(localLPP,localLP,localLX)
int localLPP;
int *localLP, *localLX;
{
   int i,j,k;
   int localLPX;
   localLPX = localLPP;
   if (localLP[0] != 1)
     {
       localLX[0] = 0;
       i = 1;
       k = 1;
       localLPX++;
     }
   else
     {
       localLP[0] = 0;
       k = 0;
       i = 0;
     }
   for (j=0;j<localLPP;j++)
     {
       localLX[i] = localLP[j];
       i++;
     }
   if (localLP[localLPP-1] != 7)
     {
       localLX[localLPP+k] = 7;
       localLPX++;
     }
   return (localLPX);
}
int acertaCP(localCPP,localCP,localCX)
int localCPP;
int *localCP, *localCX;
{
   int i,j,k;
   int localCPX;
   localCPX = localCPP;
     if (localCP[0] != 1)
     {
       localCX[0] = 0;
       i = 1;
       k = 1;
       localCPX++;
     }
   else
     {
       localCP[0] = 0;
       k = 0;
       i = 0;
     }
   for (j=0;j<localCPP;j++)
     {
       localCX[i] = localCP[j];
       i++;
     }
   if (localCP[localCPP-1] != 10)
     {
       localCX[localCPP+k] = 10;
```

```
        localCPX++;
      }
    return (localCPX);
}
```

9.2 Anexo B - código fonte da busca local

9.2.1 V1

```
void plo()
{
  static int primvez = 1;
  float fitok,fitok1,jatra;
  int i,j,k,m,lixo,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,aa,taml,tamc;
  int intcs[70];
  int mx[70];
  static  int relati,relhhh,limite,quoc;
  char linhaati[100];
  char linhahhh[100];
  FILE *fiati;
  int vezes;
  int kk;
  int colmin1,colmax1,linmin1,linmax1;
  fitok = 0.0;
  vezes = 0;
  quoc = gen / 10;
  limite = 5 + quoc;
//         ########
  while (vezes < limite) {
      vezes++;
      if (1 == 1) {  /* teste para fazer a busca local */
       UNIDA = rnd(1,10); // quando nao se quiser b. local, fazer unida = 0
//      UNIDA = 0;
       printf("... operador local %d. aplicacao - UNIDA=%d\n",vezes,UNIDA);
       fiati = fopen("relsaid4.ati","r");
       for (i=0;i<7;i++) {
         k = fscanf(fiati,"%d %d %d %d %d %d %d %d %d %d \r\n",
         &a1,&a2,&a3,&a4,&a5,&a6,&a7,&a8,&a9,&a10);
         intcs[(i*10)] = a1;
         intcs[(i*10)+1] = a2;
         intcs[(i*10)+2] = a3;
         intcs[(i*10)+3] = a4;
         intcs[(i*10)+4] = a5;
         intcs[(i*10)+5] = a6;
         intcs[(i*10)+6] = a7;
         intcs[(i*10)+7] = a8;
         intcs[(i*10)+8] = a9;
         intcs[(i*10)+9] = a10;
       }
       lixo = fclose(fiati);
       fitok = avalplo(intcs,0,vezes);
       for (i=0;i<7;i++) {
         for (j=0;j<10;j++) {
             mx[(i*10)+j] = 0;
         }
       }
       for (i=0;i<7;i++) {
         for (j=0;j<10;j++) {
                aa = intcs[(i*10)+j] + (UNIDA*(i+1) );
                if (aa < 16383) {
                   intcs[(i*10)+j] = aa;
                   fitok1 = avalplo(intcs,0,vezes);
                   if (fitok1 > fitok) {
                     mx[(i*10)+j] = +1;
```

```
                }
            }
            aa = aa - (2*(UNIDA *(i+1)));
            if (aa > 1) {
               intcs[(i*10)+j] = aa;
               fitok1 = avalplo(intcs,0,vezes);
               if ((fitok1 > fitok) && (mx[(i*10)+j] == 0)) {
                  mx[(i*10)+j] = -1;
               }
            }
            aa = aa + (UNIDA   *(i+1)     );
            intcs[(i*10)+j] = aa;
      }
   }
   for (i=0;i<7;i++)  {
     for (j=0;j<10;j++) {
         if  (mx[(i*10)+j] == +1) {
          intcs[(i*10)+j] = intcs[(i*10)+j] + (UNIDA  *(i+1)  );
         }
         if  (mx[(i*10)+j] == -1) {
          intcs[(i*10)+j] = intcs[(i*10)+j] - (UNIDA  *(i+1)  );
         }
     }
   }
   if ((relati = _creat("relsaid4.ati",0)) == -1) {
     printf("nao consigo abrir o arquivo ati de saida\n");
   }

   for (i=0;i<7;i++) {
     for (j=0;j<10;j++) {
         sprintf(linhaati,"%6d",intcs[(i*10)+j]);
         lixo=write(relati,linhaati,6);
     }
     sprintf(linhaati,"\r\n");
     lixo=write(relati,linhaati,2);
   }
   lixo = _close(relati);
   }
```

9.2.2 V2

A função é parecida, com a V1, com as seguintes modificações:

```
   if (busca_local == 6) {

   UNIDA = rnd(1,10); // quando nao se quiser b. local, fazer unida = 0
//   UNIDA = 0;
   printf("... operador local %d. aplicacao - UNIDA=%d\n",vezes,UNIDA);
   fiati = fopen("relsaid4.ati","r");    /* era ati */
   for (i=0;i<7;i++) {
     k = fscanf(fiati,"%d %d %d %d %d %d %d %d %d %d \r\n",
     &a1,&a2,&a3,&a4,&a5,&a6,&a7,&a8,&a9,&a10);
     intcs[(i*10)] = a1;
     intcs[(i*10)+1] = a2;
     intcs[(i*10)+2] = a3;
     intcs[(i*10)+3] = a4;
     intcs[(i*10)+4] = a5;
     intcs[(i*10)+5] = a6;
     intcs[(i*10)+6] = a7;
     intcs[(i*10)+7] = a8;
     intcs[(i*10)+8] = a9;
     intcs[(i*10)+9] = a10;
   }
   lixo = fclose(fiati);
   fitok = avalplo(intcs,0,vezes);
   for (i=0;i<7;i++) {
     for (j=0;j<10;j++) {
         mx[(i*10)+j] = 0;
     }
   }
```

```
      for (i=0;i<7;i++) {
        for (j=0;j<10;j++) {
              aa = intcs[(i*10)+j] + (UNIDA*(i+1) );
              if (aa < 16383) {
                 intcs[(i*10)+j] = aa;
                 fitok1 = avalplo(intcs,0,vezes);
                 if (fitok1 > fitok) {
                  fitok = fitok1;
                 }
                 else {
                  intcs[(i*10)+j] = aa - (UNIDA*(i+1));
                 }
              }
              aa = intcs[(i*10)+j] - (UNIDA*(i+1));
              if (aa > 1) {
                 intcs[(i*10)+j] = aa;
                 fitok1 = avalplo(intcs,0,vezes);
                 if (fitok1 > fitok)  {
                  fitok = fitok1;
                 }
                 else {
                  intcs[(i*10)+j] = aa + (UNIDA*(i+1));
                 }
              }
        }
      }
      if ((relati = _creat("relsaid4.ati",0)) == -1) {
        printf("nao consigo abrir o arquivo ati de saida\n");
      }

      for (i=0;i<7;i++) {
        for (j=0;j<10;j++) {
            sprintf(linhaati,"%6d",intcs[(i*10)+j]);
            lixo=write(relati,linhaati,6);
        }
        sprintf(linhaati,"\r\n");
        lixo=write(relati,linhaati,2);
      }
      lixo = _close(relati);
      }
```

9.3 Anexo C - código fonte da homogeneização

```
void homo()
{
  FILE *fiati;
  int kk,i,j,k,m,lixo,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,aa,taml,tamc;
  int intcs[70];
  int colmin1,colmax1,linmin1,linmax1;
  static  int relati;
  char linhaati[100];
  fiati = fopen("relsaid4.ati","r");
      for (i=0;i<7;i++) {
         k = fscanf(fiati,"%d %d %d %d %d %d %d %d %d %d \r\n",
         &a1,&a2,&a3,&a4,&a5,&a6,&a7,&a8,&a9,&a10);
         intcs[(i*10)] = a1;
         intcs[(i*10)+1] = a2;
         intcs[(i*10)+2] = a3;
         intcs[(i*10)+3] = a4;
         intcs[(i*10)+4] = a5;
         intcs[(i*10)+5] = a6;
         intcs[(i*10)+6] = a7;
         intcs[(i*10)+7] = a8;
         intcs[(i*10)+8] = a9;
         intcs[(i*10)+9] = a10;
      }
      lixo = fclose(fiati);
```

```
      for (kk=0;kk<3;kk++) {
      taml = fglin();
       tamc = fgcol();
      colmin1 = rnd(0,(9-tamc));
      linmin1 = rnd(0,(6-taml));
      colmax1 = colmin1+tamc;
      linmax1 = linmin1+taml;
       regula(colmin1,colmax1,linmin1,linmax1,intcs);

      }
      if ((relati = _creat("relsaid4.ati",0)) == -1) {
        printf("nao consigo abrir o arquivo ati de saida\n");
      }
      for (i=0;i<7;i++) {
        for (j=0;j<10;j++) {
            sprintf(linhaati,"%6d",intcs[(i*10)+j]);
            lixo=write(relati,linhaati,6);
        }
        sprintf(linhaati,"\r\n");
        lixo=write(relati,linhaati,2);
      }
      lixo = _close(relati);
}


void regula(colmin,colmax,linmin,linmax,intcsx)
int colmin,colmax,linmin,linmax;
int *intcsx;
{
   float fitokx,fitok1x;
   int indaux,i,j,quale,escolhe,k,achou;
   int testa[70];
   fitokx = avalplo(intcsx,0,88);
   indaux=0;
   for (i=linmin;i<=linmax;i++)  {
       for (j=colmin;j<=colmax;j++)  {
          testa[indaux] = intcsx[(i*10)+j];
          indaux++;
       }
   }
   quale=-1;
   if (indaux > 1) {
       for (escolhe=0;escolhe<indaux;escolhe++) {
          achou = 0;
          if (escolhe > 0) {
                for (k=(escolhe-1);k>=0;k--) {
                  if (testa[escolhe] == testa[k]) {
                      achou = 1;
                      k = 0;
                  }
                }
          }
          if (achou == 0) {
                for (i=linmin;i<=linmax;i++)  {
                    for (j=colmin;j<=colmax;j++)  {
                         intcsx[(i*10)+j] = testa[escolhe];
                    }
                }
                fitok1x = avalplo(intcsx,0,89);
                if (fitok1x > fitokx) {
                    quale = escolhe;
                    fitokx = fitok1x;
                }
          }
       }
   }
   if (quale != -1) {
       for (i=linmin;i<=linmax;i++) {
          for (j=colmin;j<=colmax;j++) {
             intcsx[(i*10)+j] = testa[quale];
          }
```

```
        }
    }
    else {
        indaux = 0;
        for (i=linmin;i<=linmax;i++) {
             for (j=colmin;j<=colmax;j++) {
                intcsx[(i*10)+j] = testa[indaux];
                indaux++;
             }
        }
    }
}

int fglin(void) {
    float n,sum=0;
    float bi_gauss[6];
    int i = 0;
    bi_gauss[0] = 11.0/36.0;
    bi_gauss[1] = 9.0/36.0;
    bi_gauss[2] = 7.0/36.0;
    bi_gauss[3] = 5.0/36.0;
    bi_gauss[4] = 3.0/36.0;
    bi_gauss[5] = 1.0/36.0;
    n = randomperc();
    while ((sum <= n) && (i < 6)) {
       sum = sum + bi_gauss[i];
       i++;
    }
    return i;
}

int fgcol(void) {
    float n,sum=0;
    float bi_gauss[8];
    int i = 0;
    bi_gauss[0] = 15.0/64.0;
    bi_gauss[1] = 13.0/64.0;
    bi_gauss[2] = 11.0/64.0;
    bi_gauss[3] = 9.0/64.0;
    bi_gauss[4] = 7.0/64.0;
    bi_gauss[5] = 5.0/64.0;
    bi_gauss[6] = 3.0/64.0;
    bi_gauss[7] = 1.0/64.0;
    n = randomperc();
    while ((sum <= n) && (i < 8)) {
       sum = sum + bi_gauss[i];
       i++;
    }
    return i;
}
```

10. REFERÊNCIAS BIBLIOGRÁFICAS

1. BÄCK, T. e SCHWEFEL, H.; 1993. An overvier of evolutionary algorithms for parameter optimization. Evolutionary Computation, Massachussets, v.1, 1993, p.1-23.

2. BÄCK, T., FOGEL, D. e MICHALEWICZ, Z. 1997. Handbook of Evolutionary Computation. Institute of Phyics Publishing. Bristol.

3. BÄCK, T.; 1996. Evolutionary Algorithms in Theory and Practice. 1. ed. New York: Oxford University Press.

4. BAUMER, O. 1996, "Inverting Magnetotelluric Data Using Genetic Algorithms and Simulated Annealing". *Inverse Methods: interdisciplinary elements of methodology, computation and applications*. Springer-Verlag.

5. BEASLEY, D.; BULL D.; MARTIN R, 1993a. *An overview of genetics algorithms: part 1 - fundamentals*. Inter University Committee on Computing. ftp://alife.santafe.edu/pub/USER-AREA/EC/GA/papers/over93.ps.gz.

6. BEASLEY, D.; BULL D.; MARTIN R. 1993b. *An overview of genetics algorithms: part 2 - research topics.* Inter University Committee on Computing. ftp:// alife.santafe.edu /pub /USER-AREA/EC/GA/papers/over93-2.ps.gz.

7. BITTENCOURT, G. 1996. *Inteligência Artificial - Ferramentas e Teorias.* 10º Escola de Computação, Campinas. SBC. Editora da UFSC, 1998.

8. BOSCHETTI F.; DENTITHM. C. e LIST R. D. 1996. "Inversion of Seismic Refraction Data using Genetic Algorithms". *Geophysics*. 61(6):1715-1727.

9. BRICKLE, T.; 1997. "Tournament Selection". In BÄCK, Thomas; FOGEL, David e MICHALEWICZ, Zbigniew. *Handbook of Evolutionary Computation.* release 97/1. Bristol :Institute of Physics Publishing.

10. BRUNS R. 1993. "Direct Chromosome Representation and advanced Genetic Operators for Production Scheduling". In: *Proceedings of fifth International Conference on Genetic Algorithms*. Illinois.

11. CARTWRIGHT H. M. e HARRIS S. P. 1993. "Analysis of the Distribution of Airborne Pollution using Genetic Algorithms". *Atmospheric Environment*. 27A(12):1783-1791.

12.CARTWRIGHT, H e JESSON, B. 1995. The Analysis of Waste Flow Data from Multi-unit Industrial Complexes using Genetic Algorithms. *Applied Decision Technologies Conference, Proceedings.* John Wiley.

13.DAVIS, L.; 1991. Handbook of genetic algorithms. 1. ed. New York, Van Nostrand Reinhold.

14.C de MOL, A. 1992. Critical Survey of Regularized Inversion Methods. *Inverse Problems in Scattering and Imaging.* M. Bertero e E. R. Pike, editors. London, IOP Publishing Limited.

15.DEB, K.; 1997. "Introduction to Selection". In BÄCK, Thomas; FOGEL, David e MICHALEWICZ, Zbigniew. *Handbook of Evolutionary Computation.* release 97/1. Bristol :Institute of Physics Publishing.

16.FOGEL, D. e ANGELINE, P; 1997. "Guidelines for a suitable encoding". In BÄCK, Thomas; FOGEL, David e MICHALEWICZ, Zbigniew. *Handbook of Evolutionary Computation.* release 97/1. Bristol :Institute of Physics Publishing.

17.FOGEL, D.; 1997a. "Introduction to Evolutionary Computation". In BÄCK, Thomas; FOGEL, David e MICHALEWICZ, Zbigniew. *Handbook of Evolutionary Computation.* release 97/1. Bristol :Institute of Physics Publishing.

18.FOGEL, D.; 1997b. "Principles of Evolutionary Processes". In BÄCK, Thomas; FOGEL, David e MICHALEWICZ, Zbigniew. *Handbook of Evolutionary Computation.* release 97/1. Bristol :Institute of Physics Publishing.

19.FORREST S. e MITCHELL M. 1992. "Relative Buildind-Block Fitness and the Building-Block Hypothesis". *Foundations of Genetic Algorithms 2*, D. Whitley (ed.). Morgan Kaufmann.

20.GATTO, R. C. 1993. *Um modelo de computação evolutiva para uma arquitetura cliente servidor.* São José dos Campos : INPE-Intituto Nacional de Pesquisas Espaciais. Dissertação de mestrado.

21.GOLDBERG, D. E.; 1989. Genetic Algorithms in Search, Optimization and Machine Learning. 1. ed. Reading, Massachussets: Addison Wesley Publishing Company.

22.GOODMAN E. D.; 1996. *The Genetic Algorithm Optimized for Portability and Parallelism System.* Michigan State University.

23.GUERREIRO J. N. C., BARBOSA H. J. C., GARCIA E. L. M., LOULA A. F. D. and MALTA S. M. C. 1998. "Identification of Reservoir Heterogeneities Using Tracer Breakthrough Profiles and Genetic Algorithms". *SPE Reservoir Evaluation & Engineering*, June 1998.

24.HARVEY, Inman. 1992. *Evolutionary Robotics and SAGA: the Case for Hill Crawling and Tournament Selection.* Technical Report. University of Sussex.

25.HARVEY, Inman.; 1992. Species Adaptation Genetic Algorithms: A Basis for a Continuing SAGA. Toward a Practice of Autonomous Systems, First European Conference on Artificial Life, MIT Press.

26.HEITKOETTER, J. e BEASLEY, D. eds. 1999. "The hitch-hiker's guide to evolutionary computation: a list of frequently asked questions (FAQ)", USENET:comp.ai.genetic. Available via anonymous E-mail from rtfm.mit.edu:/pub/usenet /news.answers/ai-faq/genetic/. About 110 pages.

27.HERDY, M. e PATONE, G. 1994. *Evolution strategy in action. Technical Report TR-94-05*; In: International Conference on Evolutionary Computation. Jerusalem.

28.HOFSTADTER, D. 1995. *Fluid Concepts and Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought.* New York. Basic Books.

29.HOLLAND, J. H.; 1992a. *Adaptation in Natural and Artificial Systems.* 2nd ed. Ann Harbour : The University of Michigan. (Primeira edição em 1975).

30.HOLLAND, J. H.; 1992b. Genetic algorithms. Scientific American. New York, V269, N7, July 1992, 44-50.

31.KAJIWARA I. e NAGAMATSU A. 1996. "Structural Topology Optimization by Genetic Algorithm (Approaches for Improvement of Calculation Efficiency)", %%%arrumar

32.KODIYALAM S. ; NAGGENDRA S. e DeSTEFANO J. 1996. "Composite Sandwich Structure Optimization with Application to Satellite Components". *AIAA Journal.* 54:613-622.

33.LANCZOS, C. 1961. *Linear Diferential Operators.* London. Van Nostrans Eds.

34.LORENZ, E. 1993. *A Essência do caos.* Brasília. Editora da UNB.

35.MAHFOUD, Samir W. 1995. *Niching methods for genetic algorithms.* Urbana. Doctorate Thesis - Graduate College of the University of Illinois at Urbana-Champaign.

36.MATTHEW, WALL. 1999. *Overview of Genetic Algorithms*. Tutorial in Internet at http://www-iiuf.unifr.ch/~chantem/Tutorial_on_GA

37.MICHALEWICZ, Zbiegniew. 1993. *A hierarchy of evolution programs: an experimental study. Evolutionary Computation* v.1, n.1, p. 51-76. Massachussets.

38.MICHALEWICZ, Zbiegniew. 1997. "Hybrid Methods". In BÄCK, Thomas; FOGEL, David e MICHALEWICZ, Zbigniew. *Handbook of Evolutionary Computation*. release 97/1. Bristol :Institute of Physics Publishing.

39.MINSKY, M. 1967. *Computation, Finite and Infinite Machines*. Englewood Cliffs. Prentice-Hall.

40.MITCHELL, M. 1997. *An Introduction to Genetic Algorithms*. 1st edition. Cambridge, MIT Press.

41.MITCHELL, M.; HOLLAND, J. H. e FORREST, S. 1994. *When will a genetic algorithm outperform hill climbing ?*. San Mateo : *Advances in Neural Information Processing Systems*. v.6.

42.NAG. "E04UCF Routine", 1988, *NAG Fortran Library Mark 13*, Oxford, UK.

43.NAVARRO, P.L.K.G.; de OLIVEIRA, P.P.B.; RAMOS, F.M. and VELHO, H.F.C., 1999. "Magnetotelluric inversion using problem-specific genetic operators". In: W. Banzhaf, J. Daida, A.E. Eiben, M.H. Garzon, V. Honavar, M. Jakiela, and R.E. Smith (eds.). *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*, San Francisco, CA, Morgan Kaufmann.

44.NAVARRO, P.L.K.G.; de OLIVEIRA, P.P.B.; RAMOS, F.M. and VELHO, H.F.C. 1999. *"An Evolutionary Approach in Magnetotelluric Inversion"*. In: Inverse Problems in Engineering: Theory and Practice. 3rd Int. Conference on Inverse Problems in Engineering. June 13-18, 1999 Port Ludlow, WA, USA.

45.OLIVEIRA, Pedro P. B.; RAMOS, F. M.; GATTO, R. C.; VELHO, H. F. C.; Stephany, S; NAVARRO, P. L. K. G.; HUSBANDS, P.; HARVEY, I. 1996. *A research agenda for iterative approaches to inverse problems using evolutionary computation.* In : IEEE INTERNATIONAL CONFERENCE ON EVOLUTIONARY COMPUTATION 3rd 1996, Nagoya : IEEE.

46.PENROSE, Roger. 1991. *A nova mente do rei. Computadores, mentes e as leis da física.* Rio de Janeiro, Campus.

47.RADCLIFFE, N.; 1997. "Schema processing". In BÄCK, Thomas; FOGEL, David e MICHALEWICZ, Zbigniew. *Handbook of Evolutionary Computation.* release 97/1. Bristol :Institute of Physics Publishing.

48.RAMOS, F. M; 1992: *Resolution d´un Probleme Inverse Multidimensionnel de Diffusion par la Methode des Elements Analytiques et par le Principe de l´Entropie Maximale: Contribution a la Caracterisation de Defauts Internes.* Doctorate Thesis. L´Ecole Nationale Superieure de L´Aeronautique et de L´Espace.

49.RAMOS, F.M. e CAMPOS VELHO, H.F.; 1996: "Reconstruction of Geoelectric Conductivity Distributions Using a Minimum First-Order Entropy Technique", In *2nd International Conference on Inverse Problems on Engineering*, Le Croisic, France, Vol. 2 , pp. 199-206. Publicado também no Brazilian Journal of Geophysics.

50.RAMOS, Fernando M.; GIOVANNINI, André., 1994. *Resolution d'un problème inverse multidimensionnel de diffusion de la chaleur par la méthode des éléments analytiques et par le principe de l'entropie maximale.* International Journal Heat Mass Transfer. Great Britain : v.38, n.1 p. 101-111, 1994.

51.SABATIER, P.; 1985. "Inverse Problems - an Introduction". *Inverse Problems.* Great Britain : 1:1-4.

52.SCHWEFEL, H.; 1995. *Evolution and optimum seeking.* 1. ed. New York : J. Wiley & Sons.

53.SCHWEFEL, H. 1997. "Advantages (and disadvantages) of evolutionary computation over others approaches". In BÄCK, Thomas; FOGEL, David e MICHALEWICZ, Zbigniew. *Handbook of Evolutionary Computation.* release 97/1. Bristol :Institute of Physics Publishing.

54.SCHWEFEL, H. 1997a. "Challenges to and future developments of evolutionary algorithms". In BÄCK, Thomas; FOGEL, David e MICHALEWICZ, Zbigniew. *Handbook of Evolutionary Computation.* release 97/1. Bristol :Institute of Physics Publishing.

55.STOFFA, Paul L e SEM, Mrinal K. 1991. Nonlinear multiparameter optimization using genetic algorithms: Inversion of plane-wave seismograms. *GEOPHYSICS*, (56):1794-1810.

56.TANAKA Y, ISHIGURO A e UCHIKAWA Y.; 1993. "A Genetic Algorithms Application to Inverse Problems in Electromagnetics". In: *Proceedings of fifth International Conference on Genetic Algorithms.* Illinois.

57.WOODBURY, K; 1996. *What are Inverse Problems*. Techical Report. University of Alabama.